SCHEUSE



MICRO APPLICATION

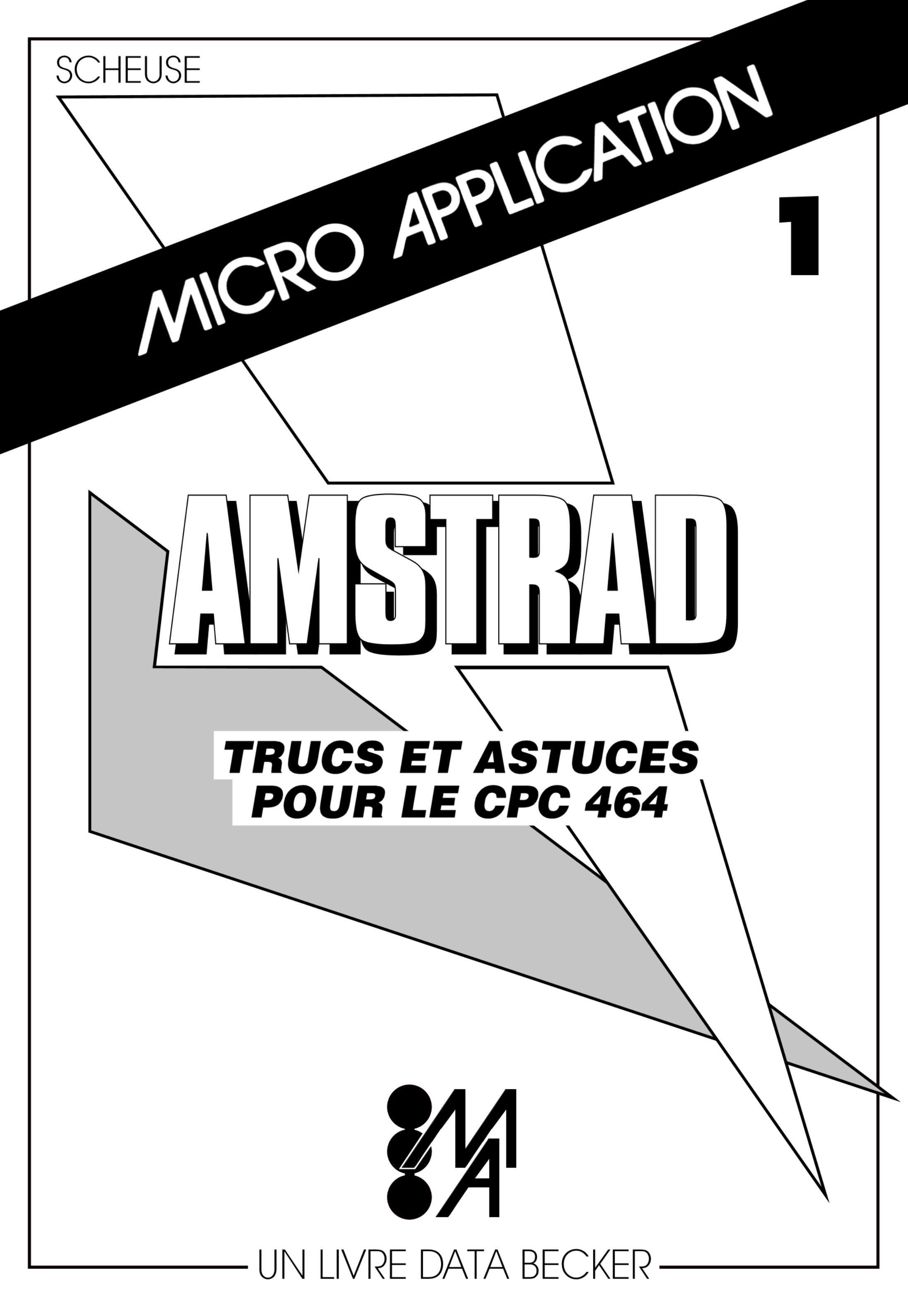
1

# AMSTRAD

## TRUCS ET ASTUCES POUR LE CPC 464



UN LIVRE DATA BECKER


SCHEUSE
MICRO APPLICATION
1
AMSTRAD
TRUCS ET ASTUCES
POUR LE CPC 464
MA
UN LIVRE DATA BECKER

Ce livre a été écrit par :

ENGLISH
GERMER
SCHEUSE
THRLIN

ISBN : 2-86899-C05-3



Traduit de l'allemand en français par Pascal Hausmann

Edité par : Frédérique BEAUDONNET
Léo BRITAN
Philippe OLIVIER

Table des matières

Chapitre 6 Routines utiles

Chapitre 7 Programmes pour l'utilisateur

# CHAPITRE 1 PREFACE

Lorsque nous avons reçu la proposition d'écrire ce livre, nous y avons répondu avec enthousiasme.

Cet enthousiasme n'a fait que grandir lorsque nous avons connu les premières données techniques de notre nouvel ordinateur. Les premières heures que nous avons passé à essayer cet ordinateur quand nous l'avons reçu ont pleinement confirmé notre attente.

Après un certain temps passé à l'utiliser, il nous est apparu évident que cet ordinateur était bien celui que beaucoup attendaient.

Il convient en effet également au lycéen, au mathématicien, au débutant, à l'expert, aux personnes intéressées par un usage commercial, aux programmeurs BASIC, aux fanas de graphisme ou de son, etc...

Son large champ d'applications se reflète d'ailleurs dans la diversité des thèmes abordés dans ce livre. Pour bien montrer qu'il ne s'agit pas de possibilités théoriques, nous avons enrichi le texte d'un grand nombre de programmes d'exemple prêts à tourner.

Le seul conseil que nous souhaitions vous donner avant que vous ne commenciez la lecture de cet ouvrage est de ne pas oublier qu'un ordinateur doit être un bon camarade et qu'il vous sera reconnaissant si vous le manipulez avec soin! En ce qui nous

concerne, nous n'avons pas eu de problème pendant tout le temps que nous nous en sommes servi.

Les auteurs

# CHAPITRE 2 LE GRAPHISME

## 2.1 INTRODUCTION

Le graphisme est un des aspects les plus fascinants d'un ordinateur. C'est pourquoi les concepteurs de votre CPC 464 ont accordé une telle importance à la partie graphique de votre ordinateur et l'ont doté des caractéristiques suivantes:

- Résolution de 640*200 points (=12800 points)
- 20, 40 ou 80 caractères sur 25 lignes
- 16, 4 ou 2 couleurs pouvant être représentées simultanément
- 32 couleurs différentes
- Instructions graphiques puissantes
- Technique de fenêtres
- Caractères auto-définis
- Caractères graphiques

C'est par le développement de leurs possibilités graphiques que les ordinateurs sont devenus les instruments universellement utilisés qu'ils sont aujourd'hui. Nous décrirons dans ce chapitre les possibilités de votre CPC dans ce domaine. Nos explications théoriques seront toujours illustrées par des exemples pratiques mais il est bien évident que le meilleur moyen pour vous de parvenir à bien assimiler les notions décrites est de réaliser vous-même de petits programmes d'exemple et de les tester.

## 2.2 LE GRAPHISME HAUTE-RESOLUTION - GRAPHIQUES A CONSULTER

La résolution est un terme qu'on emploie pour comparer les possibilités graphiques des différents ordinateurs. Pour bien comprendre sa signification, reportons-nous au temps où le papier et le crayon régnaient encore sur le monde. Il est clair que vous pouvez faire des dessins beaucoup plus précis avec un crayon fin qu'avec un crayon à mine très épaisse. De même, plus la résolution de votre ordinateur est fine, plus vous pourrez réaliser des graphiques fins et précis. La résolution est en effet le nombre de points différents que vous pouvez appeler pour les allumer ou les éteindre. Votre CPC offre une grille de 640 points horizontalement sur 200 points verticalement. Vous pouvez donc fixer 12800 points. Ceci vous permet donc réellement de réaliser de beaux graphiques: courbes de fonction, histogrammes ou graphiques camembert et bien sûr aussi des dessins pour des jeux, etc...

Vous trouverez dans la section suivante quatre programmes illustrant les possibilités graphiques de votre ordinateur. Tapez ces programmes et laissez-vous entraîner, simplement en tapant RUN dans le monde magique du graphisme. Lorsque vous vous serez lassés d'utiliser ces programmes, modifiez-les ou bien créez-en de nouveaux. Nous vous conseillons toutefois de bien lire auparavant ce chapitre en entier.

## 2.2.1 GRAPHIQUES A CONSULTER

Vous pouvez utiliser les programmes que nous vous présentons dans cette section pour créer des graphiques et les regarder ou les intégrer dans vos propres programmes.

Le premier programme dessine un cercle d'après les paramètres d'origine, de rayon et de pas. Vous pouvez choisir de faire colorier le cercle seulement, ou bien l'intérieur du cercle ou au contraire le fond de l'écran. Si le pas est trop grand, il est possible que le cercle ou le fond ne soient pas entièrement remplis.

```
10 MODE 2
20 INPUT"Origine (X)";x
30 INPUT"Origine (Y)";y
40 INPUT"Rayon (R)";r
50 INPUT"Pas (P)";s
60 INPUT"Remplir 'F'ond ou 'C'ercle";a$
70 CLS
80 z=(1 AND a$="f")+(2 AND a$="c")
90 ORIGIN x,y
100 FOR n=1 TO 360 STEP s
110 ex=ex+1
120 xp=r*COS(n)
130 yp=r*SIN(n)
140 PLOT xp,yp,1
150 IF yp=ABS(yp) THEN ey=400-y ELSE ey=y-400
160 IF xp=ABS(xp) THEN ex=640-x ELSE ex=x-640
170 IF z=1 THEN DRAW xp,ey
180 IF z=1 THEN PLOT xp,yp:DRAW ex,yp
190 IF z=2 THEN MOVE 0,0:DRAW xp,yp
200 NEXT
210 IF z<>1 THEN GOTO 280
220 FOR n=r TO 640-x
230 MOVE n,0:DRAW n,400-y
240 MOVE -n,0:DRAW -n,400-y
250 MOVE n,0:DRAW n,y-400
260 MOVE -n,0:DRAW -n,y-400
270 NEXT
280 a$=INKEY$:IF a$="" THEN 280
290 RUN
```

Exemple: Origine (x)=320

Origine (y)=200

Rayon =30

Pas =0.5

Le deuxième graphique est un pseudo-graphique en 3 dimensions. Suivant l'imagination de chacun, on peut y voir une pyramide vue d'en haut, un couloir ou une toile d'araignée. Une modification minime du programme suffit à créer une image entièrement nouvelle.

```
10 MODE 2
20 ORIGIN 320,200
30 FOR t=0 TO 180 STEP 10
40 PLOT a,t:DRAW t,-a
50 PLOT-a,t:DRAW-t,-a
60 PLOT 180,180:DRAW 0,0
70 PLOT -180,180:DRAW 0,0
80 PLOT a,t:DRAW -a,t
90 PLOT 180,-180:DRAW 0,0
100 PLOT -180,-180:DRAW 0,0
110 PLOT-t,-a:DRAW t,-a
120 a=a+10
130 NEXT t
```

Si vous modifiez par exemple la ligne 30 ainsi:

```
30 FOR T=180 TO 0 STEP -10
```

vous obtenez déjà une image très différente. Si vous faites quelques expériences avec ce programme, vous arriverez à des résultats étonnants.

Le troisième graphique est appelé caléïdoscope pour une raison que vous comprendrez dès que vous aurez lancé le programme. Ce

programme utilise une fonction de hasard qui permet d'arriver à des graphiques très intéressants.

```
10 MODE 2
20 DEF FN f(x)=INT(RND(1)*x)+1
30 FOR x=1 TO 400 STEP FN f(25)+5
40 PLOT x,0:DRAW 400,x
50 PLOT 0,x:DRAW x,400
60 PLOT x,0:DRAW 0,400-x
70 PLOT 400,x:DRAW 400-x,400
80 PLOT x,0:DRAW 400-x,400
90 PLOT 400,x:DRAW 0,400-x
100 NEXT
110 LOCATE 55,1
120 PRINT"Encore une fois?"
130 a$=INKEY$:IF a$="" THEN 130
140 IF a$="o" THEN RUN
150 END
```

Le quatrième graphique est une courbe sinusoïdale. Le programme vous demande d'indiquer le pas, puis il trace une courbe sinusoïdale non pas avec des points, mais avec des lignes qui vont d'un point (0,200) au point actuel de la courbe sinusoïdale. L'image ainsi réalisée est en pseudo 3 dimensions grâce à l'illusion optique provoquée par la superposition des lignes.

```
10 REM courbe sinusoidale
20 DEG
30 CLG
40 INPUT"Pas ";s
50 DEF FN f(x)=SIN(x)
60 FOR n=i TO 640 STEP s
70 x=x+(s*1.125)
80 MOVE 0,200
90 DRAW n,200+FN f(x)*200
100 NEXT n
110 a$=INKEY$:IF a$="" THEN 110
120 RUN
```

Essayez ce programme également avec COS(x), -COS(x) ou -SIN(x). Bien entendu, ce qu'on appelle l'art informatique n'est pas la seule application de la haute résolution de votre CPC. Vous pouvez aussi réaliser grâce à un programme votre propre blason, le faire

éditer sur une imprimante pour l'accrocher à la porte de votre château imaginaire (il y a bien sûr aussi des applications plus sérieuses).

Si vous êtes très axé sur les maths, celà vous sera certainement très utile de pouvoir faire dessiner par votre ordinateur la courbe de n'importe quelle fonction.

Si vous souhaitez un jour développer vos propres programmes (par exemple des programmes de gestion), la haute résolution vous sera très utile pour représenter par exemple les taux de croissance de votre affaire au cours des 12 derniers mois.

Toutes ces applications seraient hors de portée si votre ordinateur avait une résolution aussi faible que malheureusement la pluspart des autres microordinateurs.

## 2.2.2 FIGURES LISSAJOUS

Jules Antoine Lissajous, physicien français qui a vécu de 1822 à 1880 a étudié le mouvement de particules soumises à des mouvements périodiques. Il a découvert que les corps soumis à de tels mouvements se déplacent suivant des courbes différentes.

Nous avons écrit un programme qui vous montrera l'apparence qu'ont ces courbes.

Après avoir lancé le programme, vous devez entrer trois paramètres:

1) Fréquence Y: ce paramètre doit être compris entre 0 et 20

2) Fréquence X: ce paramètre doit être compris entre 0 et 20

3) Nombre de pas: cette valeur indique combien de points doivent être tracés.

```
10 REM figures Lissajous
20 MODE 0
30 INPUT"Frequence Y";y
40 INPUT"Frequence X";x
50 INPUT"Nombre de Pas";s
60 CLS
70 FOR a=0 TO 2*PI/s
80 PLOT 150*SIN(a*x)+150,100*COS(a*y)+100,1
90 NEXT
100 END
```

Commentaire du listing:

10 Titre

20 Fixation du mode d'écran

30 Entrée de la fréquence Y

40 Entrée de la fréquence X

50 Entrée du nombre de pas

60 Vidage de l'écran

70 Ouverture d'une boucle

80 Tracé des points

90 Fermeture de la boucle

100 Fin du programme

## 2.3 EDITEUR GRAPHIQUE

Pour utiliser ce programme, il faut que vous ayez un joystick (manche à balai) compatible avec un ATARI.
Après avoir lancé le programme, vous pouvez déplacer sur l'écran un petit point, le curseur graphique, au moyen du joystick. Après avoir appuyé sur la touche espace, vous pouvez tracer des lignes que vous pouvez effacer ensuite en appuyant d'abord sur la touche espace puis en passant sur les lignes à effacer avec le curseur graphique. Pour effacer l'écran tout entier, appuyez sur la touche 'C'. Après avoir terminé votre dessin, vous pouvez le sauvegarder en appuyant sur la touche 'S'. On vous demande alors le nom du dessin et une fois que vous l'avez entré, on vous demande d'appuyer sur les touches "REC" et "PLAY". Le dessin est alors sauvegardé bloc par bloc. La sauvegarde terminée, on vous demande si vous voulez continuer à dessiner ou arrêter le programme. Pour charger une image en dehors de l'éditeur graphique, entrez les instructions suivantes:

```
10 WINDOW 1,79,24,25:LOAD
```

Lancez ce mini-programme avec RUN et votre image sera chargée.
Ainsi, le dessin ne sera pas détruit par l'écriture et l'affichage des instructions ne se produira que dans les deux dernières lignes de l'écran.

Quand vous aurez fini d'utiliser ce programme, entrez l'instruction MODE 2 qui supprimera les fenêtres préalablement définies et effacera le dessin.

```
10 MODE 2
20 CLS
30 GOSUB 240
40 x=1:y=1:z=0
50 PLOT x,y,1
60 jo=JOY(0):IF jo<>0 THEN 80
70 a$=INKEY$:IF a$=""THEN 70
80 PLOT x,y,z
90 y=y+(1 AND jo=1)-(1 AND jo=2)
100 IF y<0 THEN y=0
110 IF y>367 THEN y=367
120 x=x+(1 AND jo=8)-(1 AND jo=4)
130 IF jo=9 THEN x=x+1:y=y+1
140 IF jo=10 THEN x=x+1:y=y-1
150 IF jo=6 THEN x=x-1:y=y-1
160 IF jo=5 THEN x=x-1:y=y+1
170 IF x<0 THEN x=0
180 IF x>639 THEN x=639
190 IF z=0 AND a$=" " THEN z=1:GOTO 210
200 IF z=1 AND a$=" " THEN z=0
210 IF a$="s" THEN GOTO 280
220 IF a$="c" THEN CLG
230 GOTO 50
240 REM
250 ORIGIN 0,33
260 WINDOW £0,1,79,24,25
270 RETURN
280 WINDOW SWAP 0
290 INPUT"Nom du dessin?",a$
300 SPEED WRITE 1
310 SAVE a$,b,49152,16383
320 PRINT"Nouveau 'D'essin ou 'F'in"
330 in$=INKEY$:IF in$="d" THEN RUN
340 IF in$<>"f" THEN GOTO 330
```

## 2.4 PLOTTER DE FONCTIONS

Le plotter de fonctions est un des auxiliaires les plus précieux des mathématiciens mais il peut aider n'importe quel lycéen dans ses devoirs de maths. Le programme suivant vous permet d'entrer une fonction qui sera plottée dès que vous aurez entré les paramètres X. Le programme vous demande en effet le minimum et le maximum X ainsi que le pas. A l'aide de ces paramètres, le maximum ou le minimum de la fonction est calculé approximativement de façon à dessiner l'axe des Y à l'échelle voulue. Ensuite la fonction est dessinée, la partie négative d'abord puis la partie positive. Le graphique terminé, le programme attend que vous appuyiez sur une touche.

La fonction utilisée crée une parabole dont la partie est disposée sur l'axe des X. Essayez les valeurs suivantes: -2 comme minimum, +2 comme maximum et 0.1 comme pas. Après un bref temps de calcul vous verrez apparaître une courbe intéressante.

Utilisez toujours la lettre X comme variable.

```
10 REM ****** PLOTTER DE FONCTIONS ******
20 MODE 2
30 CLEAR
40 DEG
50 DEF FN f(x)=ABS(x^2-1)
60 INPUT"minimum (x)";a
70 INPUT"maximum (x)";b
80 INPUT"Pas";c
90 IF a>b THEN 20
100 GOSUB 370
110 CLG
120 PLOT 0,200:DRAW 640,200:PLOT 320,0:DRAW 320,400
130 LOCATE 1,14
140 mp=320/ABS(a)
150 mq=320/ABS(b)
160 IF mp=mq THEN pr=a
170 IF mp<mq THEN pr=a
180 IF mq<mp THEN pr=-b
190 PRINT pr
200 pr$=STR$(ABS(pr))
210 LOCATE (80-LEN (pr$)),14
220 PRINT pr$
230 de$=STR$(de(1))
240 LOCATE 39-LEN(de$),1
250 PRINT de$
260 de$=STR$(-de(1))
270 LOCATE 39-LEN(de$),25
280 PRINT de$
290 mn=200/de(1)
300 FOR s=0.1 TO a STEP -c
310 PLOT 320+s*mp,200+FN f(s)*mn,1
320 NEXT
330 FOR n=0.1 TO b STEP c
340 PLOT 320+n*mq,200+FN f(n)*mn
350 NEXT
360 a$=INKEY$:IF a$="" THEN 360 ELSE 20
370 ml=(ABS(a)+ABS(b))
380 det=ml/20
390 ml=ml+2
400 DIM de(22)
410 FOR i=a TO b STEP det
420 var=var+1
430 de(var)=(FN f(i))
440 NEXT i
450 de(21)=(FN f(a)):de(22)=(FN f(b))
460 FOR j=21 TO 1 STEP -1
470 FOR i=1 TO j
480 IF ABS(de(i+1))<ABS(de(i)) THEN 520
490 de=de(i+1)
500 de(i+1)=de(i)
510 de(i)=de
520 NEXT i
530 NEXT j
540 PRINT"Maximum ou minimum probable de la fonction dans cette zone:",de(1)
550 de(1)=ABS(de(1))
560 PRINT"Appuyez sur une touche"
570 a$=INKEY$:IF a$="" THEN 570
580 RETURN
```

## 2.5 COULEURS ET CARACTERES

Les couleurs et la définition des caractères sont un autre point fort de votre CPC.

Toutefois le nombre maximum de couleurs (32) ne peut pas être obtenu dans tous les modes. Si vous ne faites pas attention à respecter cette limite, vous risquez d'obtenir une erreur IMPROPER ARGUMENT. Pour savoir combien de couleurs vous pouvez utiliser dans chaque mode, reportez-vous à votre manuel.

Voici toutefois une table de toutes les instructions pour les modes et la fixation des couleurs:

| | |
|---|---|
| Mode 0 | Efface l'écran et entre en mode 20 caractères |
| Mode 1 | Efface l'écran et entre en mode 40 caractères |
| Mode 2 | Efface l'écran et entre en mode 80 caractères |
| Border | Fixe la couleur du cadre (jusqu'à 26). Si deux couleurs sont indiquées, la couleur du cadre variera entre ces deux couleurs. |
| Ink | Affecte une couleur (second paramètre) à un canal (premier paramètre). |
| Pen | Affecte en écriture une couleur (second paramètre) à un canal (premier paramètre). |

Papier Affecte pour le fond une couleur (second paramètre) à un canal (premier paramètre).

## 2.6 INSTRUCTIONS GRAPHIQUES PUISSANTES

Nous décrirons dans ce chapitre les instructions suivantes:

PLOT et PLOTR

DRAW et DRAWR

MOVE et MOVER

ORIGIN

TEST et TESTR

XPOS et YPOS

PLOT X,Y allume le point de coordonnées absolues X,Y. La position de ce point est calculée d'après les coordonnées X et Y et l'origine actuelle. L'origine est normalement en 0,0, c'est-à-dire dans l'angle inférieur gauche de l'écran. PLOTR X,Y allume également le point de coordonnées relatives X et Y. Les coordonnées de ce point sont donc calculées à l'aide de X et Y additionnées aux coordonnées de l'emplacement actuel du curseur graphique.

DRAW X,Y est une instruction très puissante qui trace une ligne de l'emplacement actuel du curseur au point de coordonnées absolues X et Y. DRAWR X,Y trace une ligne de l'emplacement actuel du curseur au point de coordonnées relatives X et Y.

MOVE X,Y déplace le curseur, sans allumer de point, vers le point de coordonnées absolues X et Y. MOVER vers le point de coordonnées relatives X et Y.

ORIGIN permet de modifier l'origine par rapport à laquelle sont

calculées les coordonnées absolues. Cette instruction permet également de créer une fenêtre. La fenêtre originelle a une largeur de 640 points et une hauteur de 200 points.

L'instruction TEST X,Y vous permet de lire la couleur du point de coordonnées absolues X,Y, l'instruction TESTR celle du point de coordonnées relatives X,Y.

Les fonctions XPOS et YPOS vous permettent de lire l'emplacement du curseur en entrant: PRINT XPOS, YPOS.

## 2.7 LA TECHNIQUE DES FENETRES

Les fenêtres sont devenues une caractéristique des microordinateurs aussi importante que la résolution dont nous avons parlé au Chapitre 2.1.

Une fenêtre est une zone définie de l'écran de votre ordinateur que vous pouvez traiter comme un écran indépendant. L'intéressant dans la technique des fenêtres est bien sûr d'avoir la possibilité d'utiliser plusieurs fenêtres comme vous le permet votre CPC.

Une application essentielle de la technique des fenêtres est l'ergonomie. Le degré d'ergonomie mesurait à l'origine essentiellement l'adaptation matérielle d'un système informatique (ordinateur, écran, imprimante, périphériques de stockage) aux besoins des hommes qui travaillent sur ce système. Un bon degré d'ergonomie signifie qu'un travail prolongé sur un système informatique ne provoquera pas une trop grande fatigue corporelle.

Le concept d'ergonomie a été récemment étendu aux logiciels (les programmes). Les scientifiques ont en effet constaté que la présentation optique d'un programme, la présentation des données, les temps de réponse du système etc... ont une influence décisive pour rendre le travail le moins pénible possible. D'autre part on peut travailler de façon beaucoup plus efficace sur un programme bien adapté.

Pour un ordinateur tel que le CPC dont le domaine d'application privilégié serait plutôt l'informatique familiale, l'intérêt essentiel de la technique des fenêtres est la plus grande

efficacité qu'apporte cette technique, notamment dans deux types d'application très importants: les jeux et la gestion.

Les fenêtres sont très fréquemment utilisées dans les programmes de gestion, comme nous le montrerons avec notre programme de gestion de fichier au Chapitre 7.2. Les fenêtres permettent en effet d'afficher en permanence des informations dont l'utilisateur a constamment besoin (numéro d'enregistrement actuel et numéro de champ) dans la zone supérieure de l'écran. Vous pouvez également définir une fenêtre graphique avec l'instruction ORIGIN. Dans l'un des calculs de tableaux à entrer, les données sont affichées sous forme de tableaux puis restituées sous forme de figures camembert dans des fenêtres graphiques.

Les jeux sont également un domaine où la technique des fenêtres peut se révéler très utile. Vous pouvez par exemple réserver grâce à une instruction de fenêtre la zone inférieure de l'écran pour le dessin d'une route que vous n'aurez plus ensuite à modifier. Si vous programmez en BASIC, ce peut être le seul moyen d'éviter de ralentir votre jeu de façon insupportable.

Nous vous conseillons de faire plusieurs expériences pour bien assimiler la technique des fenêtres. Une première possibilité est de modifier le programme de gestion de fichier. Mais vous pouvez aussi écrire vous-même de petits programmes: essayez par exemple de créer 8 fenêtres avec 8 dessins différents ou essayez de déplacer une fenêtre sur l'écran en modifiant les valeurs des angles. Comme vous le voyez, le champ d'application des fenêtres est très vaste.

Pour vous donner quelques idées d'expériences avec les fenêtres, nous vous proposons d'essayer les deux programmes suivants. Mais il est important que vous créiez également de petits programmes utilisant les fenêtres. Même si ces programmes ne présentent en eux-mêmes aucun intérêt, l'important est que vous appreniez à bien maîtriser la technique des fenêtres pour pouvoir facilement l'intégrer ensuite à des applications plus sérieuses.

1. Le programme suivant crée 8 fenêtres qu'il remplit ensuite de caractères que vous ne pouvez pas voir parce que la couleur du fond a été modifiée en ligne 80.

Vous pouvez modifier les valeurs en lignes 500 - 510 qui correspondent aux coordonnées des fenêtres. La première ligne est la direction X (valeurs entre 1 et 80), la seconde ligne contient les valeurs Y (entre 1 et 25). Vous pouvez bien sûr créer encore plus de fenêtres en remplaçant dans les boucles la valeur 8 par le nombre de fenêtres que vous voulez.

```
10 MODE 2
20 FOR x=1 TO 8
30 READ colonne(x)
40 NEXT x
50 FOR y=1 TO 8
60 READ ligne(y)
70 NEXT y
80 PAPER 5
90 FOR x=1 TO 8
100 FOR y=1 TO 8
110 WINDOW colonne(x),colonne(x)+5,ligne(y),ligne(y)+5
120 PRINT"*********"
130 NEXT y
140 NEXT x
500 DATA 1,80,20,60,30,45,27,50
510 DATA 9,19,12,25,1,3,18,1
```

Commentaire du programme:

-------------------------

| | |
|---|---|
| 10 | Mode 80 caractères |
| 20-40 | Boucle de lecture des valeurs direction X |
| 50-70 | Boucle de lecture des valeurs direction Y |
| 80 | Fixation de la couleur du fond |
| 90-100 | Ouverture de la boucle pour X/Y |
| 110 | Création d'une fenêtre |
| 120 | Edition d'une chaîne de caractères |
| 130-140 | Fin de la boucle |
| 500-510 | Lignes de DATA |

2. Le deuxième programme remplit l'écran avec un caractère puis efface un tiers de l'écran chaque fois que vous appuyez sur une touche. Une telle routine peut être très pratique dans des programmes de gestion où l'on a souvent à effacer des parties entières de l'écran, ce qui est très facile à réaliser avec des fenêtres.

```
10 MODE 2
20 WINDOW £2,1,80,1,8
30 WINDOW £3,1,80,9,17
40 WINDOW £4,1,80,17,24
50 FOR n=2 TO 4
60 FOR mn=1 TO 639
70 PRINT"*";
80 NEXT
90 NEXT
100 FOR x=2 TO 4
110 WINDOW SWAP x
120 a$=INKEY$:IF a$="" THEN 120
130 CLS
140 NEXT
150 MODE 2
```

Commentaire du programme:

-------------------------

| | |
|---|---|
| 5 | Mode 80 caractères |
| 10-30 | Définition de 3 fenêtres |
| 35-80 | Remplissage de l'écran avec des "*" |
| 90 | Ouverture d'une boucle pour changer les fenêtres |
| 100 | Modification des fenêtres |
| 110 | Attend que vous appuyiez sur une touche |
| 120 | Effaçage d'une fenêtre |
| 130 | Fin de la boucle, on passe à la fenêtre suivante |
| 150 | Suppression des 3 fenêtres |

## 2.8 CARACTERES DEFINIS PAR L'UTILISATEUR

Les caractères définissables constituent la dernière possibilité remarquable que nous souhaitons évoquer en ce qui concerne le graphisme.

Le manuel explique malheureusement très brièvement ce que sont exactement les caractères définissables.
Un caractère se compose de points allumés ou éteints. La grille dans laquelle sont définis les caractères s'appelle une matrice. La figure 1 vous montre une matrice. Il s'agit d'une matrice 8*8. On trouve en informatique également des matrices 7*8, 7*9 et même 10*10 ou 12*12 sur les systèmes professionnels. Le CPC se trouve donc dans une bonne moyenne. Comme pour la haute résolution que nous avons traité au Chapitre 2.1, l'intérêt d'une plus grande matrice est qu'il permet une plus grande finesse de définition des caractères et donc des caractères plus lisibles.

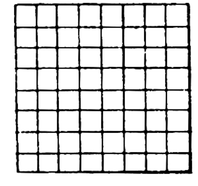


figure 1

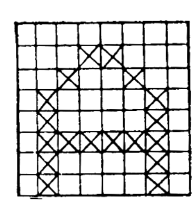


figure 2

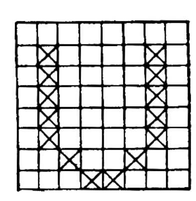


figure 3

Dans les figures 2 et 3 nous avons allumé des points pour vous montrer comment les lettres de l'écran de votre ordinateur sont créées. Le CPC dispose de 256 caractères ainsi définis dont 223 peuvent être redéfinis et donc recevoir une nouvelle forme.

Pour créer de nouveaux caractères, il faut tout d'abord en définir la forme. Le mieux pour celà est que vous dessiniez sur du papier à petit carreaux des matrices 8*8. Vous pourrez alors avec un crayon remplir des carreaux ou les effacer, jusqu'à ce que vous soyez satisfait du résultat. Pour entrer ensuite le caractère ainsi dessiné dans votre ordinateur, il faut avoir recours à l'arithmétique binaire qui est expliquée dans le Chapitre 4.

Pour le moment, reportez-vous à la figure 4. Il s'agit à nouveau d'une matrice 8*8 mais étendue puisque vous trouvez maintenant au-dessus de chaque colonne un nombre de 1 à 128.
Placez maintenant les points du caractère que vous avez dessiné dans cette matrice. Examinez maintenant la première ligne et additionnez tous les nombres qui sont placés au-dessus d'un point de la première ligne. Placez le résultat de cette addition à la droite de la ligne, comme en figure 5. Procédez de même pour les 8 lignes. Les 8 nombres que vous aurez alors calculés vont maintenant vous permettre de définir votre caractère. La figure 5 vous montre par exemple les valeurs correspondant à la lettre E.

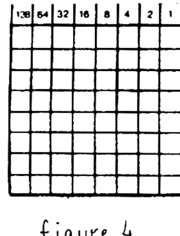
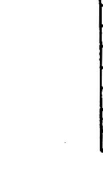



figure 4



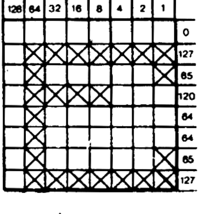



figure 5

Une fois que vous avez dessiné votre caractère dans une matrice et calculé les valeurs qui le définissent, le reste est relativement simple. Il suffit en effet de réserver de la place en mémoire pour votre caractère puis d'affecter celui-ci à une touche grâce à une instruction.

Pour réserver de la place en mémoire pour un ou plusieurs caractères, utilisez l'instruction SYMBOL AFTER. Cette instruction doit être suivie d'un paramètre entre 32 et 255. Si vous choisissez par exemple 240, vous pouvez redéfinir tous les caractères dont le code CHR$ est supérieur à 239 et vous pouvez donc redéfinir 16 caractères. Les codes CHR$ sont indiqués dans votre manuel. Plus le paramètre de l'instruction SYMBOL AFTER est

petit, plus vous occuperez de place en mémoire, ce qui ne pose d'ailleurs pas de gros problèmes avec la grande capacité mémoire du CPC 464. Si vous utilisez un paramètre plus petit que nécessaire, l'avantage est que vous pourrez définir d'autres caractères auxquels vous n'avez pas pensé au départ. Faites attention lorsque vous utilisez l'instruction SYMBOL AFTER de ne pas la réutiliser une seconde fois car ceci détruirait les caractères que vous avez déjà définis.

Les caractères redéfinis vous permettent de vous constituer votre propre jeu de caractères (futuriste, technique-mathématique, chimique, ...). Un tel jeu de caractères peut donner des résultats très intéressants si vous l'utilisez dans vos programmes de jeux. Les bons jeux sont ceux dont les acteurs du jeu ont une belle apparence. Un petit "i" bombardant de grands "B"s ne présente pas un grand intérêt ludique. Vous pouvez donc redéfinir le "i" pour lui donner la forme d'un canon et les "B"s pour leur donner la forme de dangereux extra-terrestres. Dans ce domaine vous pouvez donner libre cours à votre imagination.

Malheureusement, la méthode consistant à additionner des nombres binaires est assez pénible. C'est pourquoi vous aurez certainement beaucoup de plaisir à découvrir le programme que nous vous avons mitonné dans les pages suivantes.

## 2.8.1 UN GENERATEUR DE CARACTERES PRATIQUE

Le programme suivant vous permet de définir vos propres caractères à l'aide d'un curseur que vous déplacez dans une matrice 8*8.

Après avoir entré et lancé le programme, vous voyez apparaître un carré avec un petit point (le curseur) dans l'angle supérieur gauche. Vous pouvez déplacez le curseur avec les touches de contrôle du curseur et pour marquer un point, il vous suffit d'appuyer sur la touche 'COPY'. Le point est alors marqué par une étoile. Pour effacer un point, amenez le curseur sur la droite de l'étoile à supprimer et appuyez sur la touche espace. Pour entrer en mémoire la forme d'un caractère, appuyez sur la touche 'E'. Vous voyez alors afficher les 8 codes du caractère que vous venez de créer et on vous demande quel caractère doit être redéfini. Appuyez sur la touche à laquelle vous voulez affecter ce nouveau caractère puis appuyez sur ENTER. On vous demande ensuite si vous voulez redéfinir d'autres caractères. Tapez 'O' pour oui, 'N' pour non.

Si, après avoir mis fin au programme vous souhaitez le relancer, entrez GOTO 40 car RUN effacerait les caractères que vous avez déjà définis.

```
10 SYMBOL AFTER 32
20 IF y=0 THEN y=1
30 DIM b$(8):DIM c$(8):DIM c(8)
40 MODE 1:CLEAR
50 GOTO 240
60 x=1:y=1:z=0
70 LOCATE x,y
80 PLOT x*16,(128-y*16)+8,1
90 a$=INKEY$:IF a$="" THEN 90
100 PLOT x*16,(128-y*16)+8,0
110 z=0
120 y=y+(1 AND a$=CHR$(241))-(1 AND a$=CHR$(240))
130 IF y>8 THEN y=8
140 IF y=0 THEN y=1
150 x=x+(1 AND a$=CHR$(243))-(1 AND a$=CHR$(242))
160 IF x>8 THEN x=8
170 IF x=0 THEN x=1
180 IF a$=" "THEN z=2
190 IF a$=CHR$(224)THEN z=1
200 IF z=1 THEN PRINT"*"
210 IF z=2 THEN PRINT" "
220 IF a$="e" GOTO 320
230 GOTO 70
240 WINDOW£0, 10,17,10,18
250 ORIGIN 143,127
260 PLOT 0,0,1
270 DRAW 0,130
280 DRAW 132,130
290 DRAW 132,0
300 DRAW 0,0
310 GOTO 60
320 b$=""
330 PLOT 0,0
340 MOVER 10,120
350 FOR m=1 TO 8
360 FOR n=1 TO 8
370 IF TESTR(0,0)=1 THEN t$="1"ELSE t$="0"
380 b$(m)=b$(m)+t$
390 MOVER 16,0
400 NEXT
410 MOVE 10,120-(m*16)
420 NEXT
430 FOR x=1 TO 8:c$(x)="&X"+b$(x)
440 c(x)=VAL(c$(x)):NEXT
450 WINDOW £1,1,39,20,25
460 WINDOW SWAP 0,1
470 PRINT c(1);c(2);c(3);c(4);c(5);c(6);c(7);c(8)
480 INPUT"QUEL CARACTERE DOIT ETRE REDEFINI?",a$
490 a=ASC(a$)
500 SYMBOL a,c(1),c(2),c(3),c(4),c(5),c(6),c(7),c(8)
510 PRINT"VOULEZ-VOUS REDEFINIR D'AUTRES CARACTERES?"
520 in$=INKEY$:IF in$="" THEN 520
530 WINDOW SWAP 1,0
540 IF in$="o"THEN 40
550 MODE 1
560 PRINT"FIN DU PROGRAMME"
```

Commentaire du listing:

-----------------------

Les lignes 10-50 constituent l'initialisation du programme: dimensionnement des variables et préparation de la redéfinition des caractères. On saute ensuite à la sous-routine qui dessine la matrice. La fenêtre est un peu plus grande que la matrice pour qu'il n'y ait pas de scrolling vers le haut lorsqu'un point est marqué dans l'angle inférieur droit.

Les curseurs de texte et graphique sont alors positionnés dans l'angle supérieur gauche de la fenêtre. La ligne 100 interroge le clavier et les lignes 120 - 220 évaluent le résultat. Si vous appuyez sur la touche 'E' (ligne 220) on saute au calcul des valeurs composant le caractère: on teste d'abord si au moins un

point figure dans une ligne (ligne 370). Si un point est mis, on ajoute un 1 à la variable B$(M), sinon on ajoute un 0. On constitue ainsi pour chaque ligne un nombre binaire de 8 chiffres. Les lignes 430 et 440 transforment les nombres binaires en nombres décimaux et les placent dans la variable C(W). Une fenêtre de texte est alors créée dans laquelle les codes sont affichés et les questions vous sont posées. La ligne 500 introduit le caractère que vous avez créé dans le jeu de caractères. L'instruction MODE 1 supprime alors toutes les fenêtres.

Pour vous montrer ce qu'on peut faire avec un programme générateur de caractères, nous avons écrit un petit programme qui vous permet de créer les accents français aigu et grave. Voici tout d'abord une table des codes de ces accents et des touches auxquelles nous les avons affectés:

| Accent | Touche | Numéro de touche | Codes |
|---|---|---|---|
| à | : | 58 | 32,16,120,12,124,204,118,0 |
| é | * | 42 | 8,16,60,102,126,96,60,0 |
| è | ; | 59 | 16,8,60,102,126,96,60,0 |
| ù | + | 43 | 32,16,102,102,102,102,62,0 |

Voici maintenant le programme. Après l'avoir lancé, les accents sont définis et l'écran est vidé. Une dernière remarque: même lorsque la forme d'une touche a été redéfinie, sa fonction reste la même.

```
10 FOR N=1 TO 4
20 FOR M=1 TO 9
30 READ A(M)
40 NEXT M
50 SYMBOL A(1),A(2),A(3),A(4),A(5),A(6),A(7),A(8),A(9)
60 NEXT N
70 CLS
80 DATA 58,32,16,120,12,124,204,118,0
90 DATA 42,8,16,60,102,126,96,60,0
100 DATA 59,16,8,60,102,126,96,60,0
110 DATA 43,32,16,102,102,102,102,62,0
```

Ces accents vous seront certainement très utiles si vous voulez par exemple écrire des lettres avec votre CPC. D'autre part celà constituera un excellent exercice pour vous de définir également les accents circonflexes et la cédille.

Voici maintenant encore un programme qui travaille avec des caractères redéfinis qui vous montrera bien que cette technique permet (presque) d'atteindre la qualité des dessins animés.

Le programme vous montre une auto fonçant sur un camion à l'arrêt. Ce mouvement peut être créé simplement avec une boucle et l'instruction LOCATE.

Essayez ensuite de développer ce programme. Vous pouvez par exemple modifier la route que suit l'auto ou faire bouger également le camion. Vous pouvez également créer des caractères permettant de suivre la déformation de la voiture. Et laissez le champ libre à votre imagination!

```
5 MODE 1
10 SYMBOL AFTER 32
20 SYMBOL 91,1,2,4,8,255,255,255,63
30 SYMBOL 93,252,36,36,36,255,255,255,24
40 SYMBOL 33,127,127,127,127,127,79,15,7
60 SYMBOL 35,255,255,255,255,255,190,190,28
70 SYMBOL 36,255,255,255,255,255,124,0,0
80 SYMBOL 37,255,255,255,255,255,251,248,112
90 SYMBOL 38,127,127,127,127,127,127,127,127
100 SYMBOL 39,255,255,255,255,255,255,255,255
105 SYMBOL 40,255,255,255,255,255,255,255,255
110 SYMBOL 41,240,240,240,240,255,241,241,241
111 LOCATE 2,13:PRINT"&'')"
112 LOCATE 2,14:PRINT"!#$%"
120 FOR x=39 TO 6 STEP -1
130 LOCATE x,14:PRINT CHR$(91);CHR$(93)
135 LOCATE x,14:PRINT"  "
140 NEXT x
150 GOTO 150
```

Commentaire du programme:

------------------------

| | |
|---|---|
| 5 | Fixation du mode |
| 0-110 | Définition du caractère |
| 111 | Affichage de la partie supérieure de la voiture |
| 112 | Affichage de la partie inférieure de la voiture |
| 120 | Ouverture d'une boucle pour le déplacement de la voiture dans la direction X |
| 130 | Affichage du véhicule |
| 135 | Effaçage de l'ancien véhicule |
| 140 | Fin de la boucle, la voiture est déplacée vers la gauche |

## 2.9 CARACTERES GRAPHIQUES

Les caractères graphiques sont une autre possibilité très intéressante de votre CPC. Les caractères graphiques sont des caractères que vous ne pouvez obtenir simplement en frappant une touche mais uniquement en utilisant la fonction CHR$(code). Les codes de tous les caractères graphiques prédéfinis de votre CPC ainsi que leur représentation dans une matrice se trouvent à la fin de votre manuel.

Pour faire apparaître par exemple une bombe à l'écran, il faut entrer PRINT CHR$(252).

Vous pouvez faire éditer de cette façon tous les caractères graphiques. Vous pouvez également les déplacer avec LOCATE:

```
10 MODE 2
20 FOR Y=1 TO 25
30 LOCATE 40,Y:PRINT CHR$(252)
40 LOCATE 40,Y:PRINT" "
50 NEXT Y
```

Ce petit programme fait tomber la bombe à travers l'écran avec les instructions LOCATE et PRINT.

La même technique peut vous permettre de réaliser des animations, notamment pour vos jeux, qui peuvent être très spectaculaires avec un peu d'adresse.

Voici maintenant un autre programme d'exemple qui vous montrera

avant tout la vitesse de cette méthode. Le programme se termine par un message d'erreur mais ceci n'est pas très grave: peut-être pourriez-vous essayer de trouver pourquoi. Un conseil: faites afficher à la fin du programme la valeur de Y; vous devriez alors comprendre d'où vient l'erreur.

```
10 MODE 2
20 FOR x=1 TO 25
30 t=36
40 q=12
50 LOCATE t,x:PRINT STRING$(&8,CHR$(249))
60 NEXT x
70 y=1
80 FOR x=q TO q+4
90 LOCATE y,x:PRINT STRING$(&8,CHR$(250))
100 NEXT x
110 FOR x=q TO q+4
120 LOCATE y,x:PRINT STRING$(&8," ")
130 NEXT x
140 y=y+8
150 GOTO 80
```

Commentaire du programme:

------------------------

| | |
|---|---|
| 10 | Passage en mode 80 caractères |
| 20 | Ouverture de la boucle du mouvement descendant |
| 30 | Valeur initiale pour le mouvement descendant des petits hommes |
| 40 | Valeur Y pour les petits hommes fixes |
| 50-60 | Boucle d'édition des petits hommes verticaux |
| 70 | Test si Y égale 1 |
| 80-100 | Boucle pour le mouvement horizontal avec des pas de 4 |
| 110-130 | Boucle pour supprimer le dernier bloc de petits hommes |
| 140 | Incrémentation de la valeur pour que les petits hommes se déplacent vers l'avant par blocs |

## 2.10 LA MEMOIRE ECRAN

La mémoire principale de votre CPC a une taille de 64K, soit 65536 octets. Vous pouvez utiliser pour vos programmes BASIC environ 43K.

La mémoire est organisée en 4 blocs de 16K. Ces blocs sont appelés banques et sont numérotés de 0 à 3. La banque 0 est donc le bloc allant de l'adresse 0 à l'adresse 16383, la banque 3 le bloc de 49152 à 65535. Lorsque vous allumez votre ordinateur, la mémoire-écran occupe la banque 3. Quel que soit donc le mode de gestion de l'écran que vous utilisez, il y a toujours 16K qui sont réservés pour la mémoire-écran. Entrez maintenant les lignes suivantes et lancez le programme avec RUN:

```
10 REM Ecran 1
20 MODE 2:PRINT"a"
30 FOR t=&C000 to &FFFF STEP &800
40 PRINT BIN$(PEEK(t),8):NEXT t
```

Si vous ne vous êtes pas trompé, vous voyez apparaître les informations suivantes:

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 1 1 1 0 0 0
0 0 0 0 1 1 0 0
0 1 1 1 1 1 0 0
1 1 0 0 1 1 0 0
0 1 1 1 0 1 1 0
0 0 0 0 0 0 0 0
```

Vous reconnaissez la carte matricielle du a minuscule que la ligne 10 a fait afficher dans l'angle supérieur gauche de l'écran. Modifiez quelque peu votre programme:

```
10 REM Ecran 2
20 FOR t=&C000 to &FFFF STEP &800
30 FOR a=0 TO 3
40 PRINT BIN$(PEEK(t+a),8);" ";
50 NEXT a:PRINT
60 NEXT t
```

Entrez maintenant MODE 2 en mode direct puis RUN. Vous obtenez les cartes matricielles des lettres R e a d. En effet après avoir exécuté l'instruction MODE 2, votre ordinateur a écrit Ready en haut de l'écran.
Vous pouvez maintenant essayer de voir ce qui se produira si vous choisissez un autre mode avant de lancer le programme. Entrez MODE 1 puis RUN. Le résultat est inattendu. Vous pouvez encore distinguer un R et un e mais vous voyez que la carte de chaque lettre est répartie sur deux matrices et la moitié droite de chaque matrice est pleine de 0. Essayons de savoir pourquoi les lettres apparaissent deux fois plus grosses en mode 1 bien qu'en fait il n'y ait pas plus de 1 qu'en mode 80 caractères.

Entrez:

```
PEN 2
MODE 1
RUN
```

Vous pouvez à nouveau reconnaître les deux lettres R et e, toujours réparties chacune sur deux matrices. Mais la différence

est que c'est maintenant la partie gauche de chaque matrice qui est constituée de 0 alors que les bits allumés se trouvent dans la partie droite.

La raison de cette différence tient à l'organisation de la mémoire-écran de 16K. En effet les mêmes bits sont utilisés à la fois pour définir la forme mais aussi les couleurs des caractères. Il faut maintenant examiner comment ceci se passe exactement suivant le mode de l'écran qui a été choisi.

2.10.1 MODE 2

En mode 2, les choses restent relativement simples. Chaque bit mis (=1) correspond à un pixel (mini-point de l'écran) allumé avec la couleur des caractères. Un 1 signifie donc qu'un pixel reçoit la couleur bleue des caractères (définie par le stylo: PEN) et un 0 qu'il reçoit la couleur jaune du fond de l'écran (définie par le papier: PAPER). Par exemple, si vous entrez:

```
MODE 2:PAPER 1:PEN 0
CLS
```

et que vous lancez Ecran 2 avec RUN, vous voyez que chaque 1 correspond toujours à un point jaune qui est maintenant la couleur du fond de l'écran et chaque 0 toujours à un point bleu, nouvelle couleur des caractères.

En résumé: en MODE 2, chaque bit de la mémoire-écran indique la couleur du point correspondant de l'image de l'écran:

0 = bleu

1 = jaune

## 2.10.2 MODE 1

Passez maintenant en MODE 1. Ce mode vous permet d'avoir 40 caractères par ligne et de définir 4 couleurs différentes pour les caractères. Chaque octet de la mémoire-écran indique quelle couleur doit avoir un point de l'écran.
Examinons un de ces octets:

```
        ----------------------------------
Bit No  I 7 I 6 I 5 I 4 II 3 I 2 I 1 I 0 I
        ----------------------------------
```

En MODE 1, chaque octet est divisé en deux quartets, comprenant donc 4 bits chacun, que nous avons délimités par un II. Les bits occupant la même position dans leurs quartets respectifs constituent une paire: bits 3 et 7, 2 et 6, 1 et 5, et 4 et 0. Chacun de ces couples de bits représente donc une des quatre couleurs disponibles pour les caractères. On ne pourrait donc en principe fixer de cette façon les couleurs que de 4 pixels de l'écran seulement. Comme chaque octet doit correspondre à 8 pixels, chaque couple de bits correspond en fait également à deux pixels contigus. Les bits 3 et 7 correspondent à la paire de pixels située à l'extrême gauche.
Voici une table des couleurs ainsi représentées par chaque couple de bits:

| Valeur des bits | Couleur |
|---|---|
| 0 0 | Bleu |
| 0 1 | Jaune clair |

| | |
|---|---|
| 1 0 | Bleu-vert clair |
| 1 1 | Rouge clair |

Il est maintenant temps de se reporter au programme "Ecran 2" et aux deux exemples que nous vous demandions d'essayer: la première ligne de la première carte matricielle se présentait ainsi:

1 1 1 1 0 0 0 0

Reconstituez maintenant les couples de bits solidaires et regardez dans la table à quelle couleur ils correspondent:

| | | | | |
|---|---|---|---|---|
| Couple de bits: | 3,7 | 2,6 | 1,5 | 0,4 |
| Valeur: | 0 1 | 0 1 | 0 1 | 0 1 |
| Couleur: | jaune | jaune | jaune | jaune |

Pour bien comprendre le système de mise en mémoire des couleurs en MODE 1, prenez une feuille de papier et tracez-y 4 carrés de 8 carreaux sur 8. Divisez chaque carré en 4 colonnes de deux carreaux de large et en 8 lignes. Chaque paire de carreaux sur votre papier correspond à une paire de pixels à l'écran. Remplissez chaque paire de carreaux si le couple de bits correspondant indique la couleur jaune. Laissez-la vide si les bits indiquent bleu. Dans notre exemple vous ne devez pas obtenir de couleur autre que jaune ou bleu. Quand vous aurez terminé, vous obtiendrez un R et un e très larges.
Changez maintenant la couleur des caractères avec l'instruction PEN 2. Entrez CLS et RUN. Vous constatez maintenant que tous les couples de bits contiennent soit 10 soit 00. En consultant la table, vous voyez que 10 produit la couleur bleu-vert clair.

Après toutes ces petites expériences, il ne devrait plus vous être très difficile de réaliser maintenant des caractères de quatre couleurs différentes. Notez cependant que de tels caracteres ne peuvent pas être entrés en memoire dans la table des caractères avec les instructions SYMBOL ou SYMBOL AFTER. Le seul moyen de les réaliser est d'en POKEr les valeurs dans la mémoire-ecran. Le mieux est donc de placer ces valeurs dans des lignes de DATA qui seront placées par le programme dans la mémoire-écran.
Vous trouverez à la fin de cette section la formule vous permettant de calculer dans quel emplacement de la mémoire-ecran vous devez POKEr un caractere de quatre couleurs.

Ce mode de fonctionnement de votre CPC permet à votre ordinateur d'atteindre sa puissance maximum dans le domaine des couleurs, au détriment toutefois de la résolution. En effet, pour pouvoir sélectionner pour chaque pixel 16 couleurs différentes, il faudrait une mémoire de 64K. Comme c'est déjà le cas en mode 1, le mode 0 a recours au petit truc qui consiste à affecter à un groupe de pixels contigus la même couleur. En mode multicolore, chaque couleur est fixée pour un groupe de 4 points contigus. Voici la table des couleurs disponibles:

| Bin | Hex | Déc | Couleur |
|---|---|---|---|
| 0000 | 0 | 0 | Bleu |
| 0001 | 1 | 1 | Jaune clair |
| 0010 | 2 | 2 | Bleu-vert clair |
| 0011 | 3 | 3 | Rouge clair |
| 0100 | 4 | 4 | Blanc brillant |
| 0101 | 5 | 5 | Noir |
| 0110 | 6 | 6 | Bleu clair |
| 0111 | 7 | 7 | Magenta clair |
| 1000 | 8 | 8 | Bleu-vert |
| 1001 | 9 | 9 | Jaune |
| 1010 | A | 10 | Bleu pastel |
| 1011 | B | 11 | Rose |
| 1100 | C | 12 | Vert clair |
| 1101 | D | 13 | Vert pastel |
| 1110 | E | 14 | Jaune vif |
| 1111 | F | 15 | Bleu ciel |

Essayons de placer ces 16 couleurs à l'écran:

```
10 REM Exemple de couleurs
20 DATA &00,&11,&22,&33,&44,&55,&66,&77,&88,&99,&AA,&BB,
        &CC,&DD,&EE,&FF
30 MODE 0
40 FOR d=0 TO 31 STEP 2
50 READ x
60 FOR t=&C000 TO &FFFF STEP &800
70 POKE t+d,x
80 NEXT t:NEXT d
90 GOTO 90
```

Si vous lancez ce programme avec RUN vous n'obtiendrez pas le résultat voulu. Comme en mode 1, les bits qui servent à indiquer une couleur ne sont pas des bits contigus: la couleur des 4 pixels les plus à gauche est indiquée par les bits 1,3,5 et 7 d'un octet, celle des 4 pixels les plus à droite par les bits 0,2,4 et 6. D'autre part l'ordre dans lequel chaque groupe de 4 bits doit être considéré est le suivant:

4 pixels de gauche: bits 1,5,3,7

4 pixels de droite: bits 0,4,2,6

Nous pouvons maintenant appeler toutes les couleurs disponibles:

| Bit No | 7 6 5 4 3 2 1 0 | | gauche | droite | octet |
|---|---|---|---|---|---|
| | 0 0 0 0 0 0 0 0 | = | 0 | 0 | 0 |
| | 1 1 0 0 0 0 0 0 | = | 1 | 1 | 192 |
| | 0 0 0 0 1 1 0 0 | = | 2 | 2 | 12 |
| | 1 1 0 0 1 1 0 0 | = | 3 | 3 | 204 |
| | 0 0 1 1 0 0 0 0 | = | 4 | 4 | 48 |
| | 1 1 1 1 0 0 0 0 | = | 5 | 5 | 240 |
| | 0 0 1 1 1 1 0 0 | = | 6 | 6 | 60 |
| | 1 1 1 1 1 1 0 0 | = | 7 | 7 | 252 |
| | 0 0 0 0 0 0 1 1 | = | 8 | 8 | 3 |
| | 1 1 0 0 0 0 1 1 | = | 9 | 9 | 195 |
| | 0 0 0 0 1 1 1 1 | = | 10 | 10 | 15 |
| | 1 1 0 0 1 1 1 1 | = | 11 | 11 | 207 |
| | 0 0 1 1 0 0 1 1 | = | 12 | 12 | 51 |
| | 1 1 1 1 0 0 1 1 | = | 13 | 13 | 243 |
| | 0 0 1 1 1 1 1 1 | = | 14 | 14 | 63 |
| | 1 1 1 1 1 1 1 1 | = | 15 | 15 | 255 |

Il faut donc modifier le programme Exemple de couleurs en tenant compte des valeurs indiquées dans la colonne octet:

```
20  DATA 0,192,12,204,48,240,60,252,3,195,15,207,51,243,
        63,255
```

Si vous lancez maintenant le programme ainsi modifié, vous obtenez bien les 16 couleurs possibles, dans le bon ordre. Vous ne distinguez bien sûr que 15 couleurs puisque la couleur 0 est identique à la couleur du fond de l'écran.

## 2.10.4 POSITIONNEMENT D'UN CARACTERE

Dans les derniers chapitres, nous avons souvent parlé des caractères multicolores ainsi que de la façon de les réaliser. Voici maintenant un petit programme utilitaire qui vous permettra de positionner une carte mémoire de couleurs sur l'écran:

```
10 REM Positionnement d'un caractère
20 MODE 0
30 INPUT"Quelle colonne";c
40 LOCATE 1,1
50 INPUT"Quelle ligne";l
60 LOCATE 1,1
70 INPUT"Code couleur ";couleur
80 LOCATE 1,1
90 base=&C000
100 FOR adresse=base TO base + &3FFF STEP &800
110 POKE adresse + (c-1) + ((l-1)*80),couleur
120 NEXT adresse
130 GOTO 30
```

Ce programme POKE des petits carrés de la taille du curseur en mode 2 dans l'emplacement que vous définissez par ses numéros de colonne et de ligne. Le code couleur qui vous est demandé n'est pas identique à celui que vous trouvez dans votre manuel. Il faut que vous calculiez ce code vous-même, ce qui ne devrait pas être trop difficile si vous avez lu attentivement le chapitre 2.10.3. La ligne 90 affecte à la variable base la valeur standard pour le début de la mémoire écran. Vous trouverez cependant au Chapitre 6 une routine en langage-machine pour déplacer le début de la

mémoire-écran. Si vous utilisez cette routine, il faudra ensuite modifier la ligne 90 en fonction de la nouvelle adresse de début de la mémoire-écran.

La ligne 110 est la ligne la plus importante du programme puisqu'elle place le carré de couleur dans l'emplacement voulu. Notez que l'angle supérieur gauche de l'écran (position HOME) a pour coordonnées 1,1 et qu'il n'y a pas dans la forme actuelle du programme de routine contrôlant la validité des valeurs entrées.

STEP &800

Il ne vous a certainement pas échappé que dans toutes les boucles de lecture ou d'écriture d'un caractère dans la mémoire-écran apparaît le pas "STEP &800". C'est qu'en effet, contrairement à la plupart des autres ordinateurs où les 8 octets correspondant aux lignes composant un caractère sont généralement disposés à la suite les uns des autres, dans votre CPC sont tout d'abord placées à la suite les unes des autres les premières lignes des 2000 emplacements possibles de l'écran (25 lignes de 80 caractères), puis les secondes lignes de ces 2000 emplacements, etc...
L'adresse de la première ligne du premier emplacement est aussi l'adresse du début de la mémoire-écran: &C000 = 49152. La seconde ligne du premier emplacement de l'écran se trouve en &C800 = 51200, etc... C'est pourquoi, une fois calculée l'adresse en mémoire de la ligne d'un caractère, il faut toujours ajouter &800 à cette position pour obtenir l'adresse de la ligne suivante du même caractère:

| | Hex | Décimal |
|---|---|---|
| Ligne 1 | C000 | 49152 |
| Ligne 2 | C800 | 51200 |
| Ligne 3 | D000 | 53248 |
| Ligne 4 | D800 | 55296 |
| Ligne 5 | E000 | 57344 |
| Ligne 6 | E800 | 59392 |
| Ligne 7 | F000 | 61440 |
| Ligne 8 | F800 | 63488 |

Chaque ligne utilise donc un bloc de &800 = 2048 octets. Or il n'y a que 2000 emplacements différents sur l'écran. Celà signifie que dans chacun de ces blocs, 48 octets sont inutilisés dans lesquels vous pouvez placer des données ou des routines en langage-machine, sauf toutefois si vous utilisez la possibilité de scrolling horizontal que nous aborderons au Chapitre 6. Dans ce cas en effet ces octets sont utilisés pour la gestion de l'écran.

## 2.11 LES CARACTERES ALTERNATIFS VUS SOUS UN AUTRE ANGLE

Nous vous avons déjà indiqué au chapitre 2.8 une manière de modifier le jeu de caractères. Il peut cependant arriver que la technique fondée sur les instructions SYMBOL et SYMBOL AFTER ne soit plus satisfaisante, soit que vous ne souhaitiez créer qu'un nouveau caractère, soit que vous travailliez entièrement en langage-machine, ou bien enfin que vous ne vouliez modifier que 4 ou 5 points d'un caractère existant. Dans ce cas nous vous conseillons d'utiliser la méthode à laquelle fait appel le programme suivant:

```
10 REM Lire le jeu de caractères
20 MODE 1
30 FOR t=&AB80 TO &AB87
40 PRINT BIN$(PEEK(t),8)
50 NEXT t
```

Vous obtenez à l'écran la carte bits de la flèche pointée vers le haut. Le code de ce caractère est 240. Vous pouvez lire de la même façon tous les caractères dont le code est supérieur à 239. Les choses sont plus compliquées pour les autres caractères. C'est pourquoi nous nous en tiendrons là pour le moment. Il est de même très facile de modifier ces 16 caractères. Essayez la ligne BASIC suivante:

```
POKE &AB87,255
```

Lancez à nouveau le programme ci-dessus qui doit toujours se trouver en mémoire et vous verrez que notre flèche a maintenant un

"pied". Pour éditer la flèche à l'écran, entrez:

```
PRINT CHR$(240):PRINT
```

La seconde instruction PRINT vous permet de bien voir le nouveau caractère sans être gêné par le mot Ready.

Voici la liste des caractères auxquels vous pouvez accéder librement pour les lire ou les modifier, en suivant la même méthode:

| Hex de | Hex à | Décimal de | Décimal à | CHR$( ) |
|---|---|---|---|---|
| AB80 | AB87 | 43904 | 43911 | 240 |
| AB88 | AB8F | 43912 | 43919 | 241 |
| AB90 | AB97 | 43920 | 43927 | 242 |
| AB98 | AB9F | 43928 | 43935 | 243 |
| ABA0 | ABA7 | 43936 | 43943 | 244 |
| ABA8 | ABAF | 43944 | 43951 | 245 |
| ABB0 | ABB7 | 43952 | 43959 | 246 |
| ABB8 | ABBF | 43960 | 43967 | 247 |
| ABC0 | ABC7 | 43968 | 43975 | 248 |
| ABC8 | ABCF | 43976 | 43983 | 249 |
| ABD0 | ABD7 | 43984 | 43991 | 250 |
| ABD8 | ABDF | 43992 | 43999 | 251 |
| ABE0 | ABE7 | 44000 | 44007 | 252 |
| ABE8 | ABEF | 44008 | 44015 | 253 |
| ABF0 | ABF7 | 44016 | 44023 | 254 |
| ABF8 | ABFF | 44024 | 44031 | 255 |

# CHAPITRE 3 LE SON

## 3.1 INTRODUCTION

Le chapitre suivant explique quelques principes fondamentaux de la production de bruits et de sons:

- connexion du CPC 464 sur une chaîne stéréo
- principes de la musique et du son
- explication des instructions de production de sons
- env (enveloppe du volume)
- ent (enveloppe du ton)
- éditeur de sons
- démonstration d'applications dans des jeux

Nous avons illustre autant que possible par des exemples concrets les notions un peu théoriques qui vont suivre. Lisez donc le chapitre suivant avec votre ordinateur et n'hésitez pas à taper les programmes les plus importants.

Si vous ne comprenez pas très bien une notion quelconque relisez attentivement le passage de votre manuel qui s'y rapporte.

Si vous avez l'esprit plus mathématique que musical, vous vous apercevrez bien vite que la musique, notamment la musique sur ordinateur est très liée aux mathématiques et à la physique. Si tout celà vous semble trop compliqué, vous pourrez au moins prendre plaisir à écouter les jolies mélodies dont vous trouverez les listings à la fin de ce chapitre.

## 3.2 CONNEXION DU CPC SUR UNE CHAINE STEREO

Vous avez certainement déjà remarqué que le haut-parleur intégré ne rend pas justice aux possibilités étendues de votre CPC dans le domaine musical. C'est pourquoi nous vous expliquons en détail comment brancher votre ordinateur sur une chaîne stéréo.

Si vous essayez de brancher directement un casque sur la connexion se trouvant à l'arrière de votre CPC, vous constaterez que celui-ci ne produit aucun son par lui-même car le signal qu'il sort n'est pas amplifié et n'est donc pas audible avec votre casque.

Il faut donc utiliser un amplificateur tel qu'en possède toute chaîne stéréo ou bien sûr également tout appareil de radio ou lecteur de cassette.

Comment allez-vous envoyer le signal à votre amplificateur? Réalisez pour celà un câble de branchement en dérivé avec du côté relié à votre CPC une fiche mâle de 3.5 mm de diamètre. Notez bien qu'il s'agit d'une transmission tripolaire comme pour un casque de walkman par exemple. Du côté de votre amplificateur, il vous faut une autre fiche mâle pouvant s'enficher dans l'entrée microphone. C'est en effet l'entrée micro qui réagit le mieux à des sons qui n'ont pas été amplifiés préalablement. Si vous ne disposez pas d'une telle entrée sur votre amplificateur, utilisez l'entrée pour la platine disque ou pour le magnétophone.

Reliez maintenant les deux appareils ensemble, non sans les avoir éteints tous deux au préalable pour éviter des complications très désagréables.

Après avoir relié les deux appareils puis après les avoir à nouveau allumés, baissez complètement le volume pour éviter une surprise désagréable.

Vous pouvez maintenant tester le volume le plus adapté en entrant l'instruction suivante:

```
SOUND 7,284,32767,7
```

Cette instruction produit le LA 440 qui est utilisé maintenant par les musiciens du monde entier pour accorder leurs instruments. Toutes les autres notes sont calculées à partir de celle-ci.

## 3.3 PRINCIPES DE BASE DU SON

La production de bruits ou de sons est un aspect très important des possibilités d'un microordinateur familial. Dans ce domaine, le CPC 464 peut par ses caractéristiques remarquables supporter la comparaison avec les meilleurs ordinateurs de sa catégorie.

Le CPC peut par exemple produire des ultra-sons, c'est-à-dire des sons que vous n'entendrez pas parce qu'ils dépassent le spectre sonore audible par une oreille humaine, mais que votre chien par contre pourra entendre. En effet, un homme peut entendre des sons compris entre 16 Hz et 16 Khz alors que le CPC peut produire des sons entre 30 Hz et 125 Khz. Si par exemple vous entrez SOUND 1,1,100 vous n'entendrez rien alors que votre chien réagira certainement.

Vous savez qu'un morceau de musique n'est qu'une suite de tons de différentes fréquences. Si la succession des tons choisis est heureuse, vous aurez de la belle musique, dans le cas contraire de la mauvaise ou, du moins, ce que vous considérez comme telle.

La belle musique ne sonne cependant bien que si vous disposez également d'un bel instrument, comme c'est le cas avec le CPC 464.

Voyons maintenant comment se produit un son pour mieux comprendre le principe de la musique sur ordinateur:

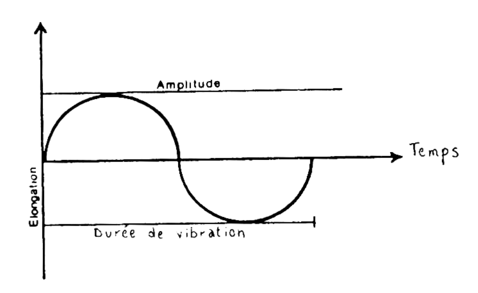
Un son se compose d'ondes ou vibrations. Un ton est plus haut qu'un autre si sa fréquence est supérieure, donc s'il se compose de plus de vibrations dans un même intervalle de durée. L'unité de

mesure des fréquences est le Hertz (Hz).

Voici un exemple:

Le do du milieu (do de la clé sur un piano) a une fréquence de 261,626 Hz.
Le la de référence international a une fréquence de 440 Hz et est donc plus haut que le do du milieu.

La figure suivante représente une vibration avec tous les termes importants que nous allons maintenant expliquer.



1. Elongation(écart momentané par rapport à la situation de pause)
---

Ce concept n'est pas très important pour la production de sons car il ne peut pas être employé d'une manière pratique. Il s'agit simplement du nom de l'axe des Y.

2. Amplitude (écart maximum ou amplitude de vibration)
---

Cette valeur correspond au volume d'un son: plus l'amplitude est grande, plus le son est fort.

3. Durée
---

Ce facteur jouera un grand rôle pour la constitution des courbes d'enveloppe. Il correspond à l'axe des X de notre graphique.

4. Durée de vibration
---

Il s'agit de la durée d'une vibration complète. Ce facteur est inversement proportionnel à la fréquence.

Pour ceux qui ne connaissent pas déjà tous les secrets de la physique des vibrations voici le principe qui permet à votre CPC de produire des sons:

Un haut-parleur, qui est indispensable pour que votre ordinateur puisse produire des sons, se compose d'un aimant, d'une bobine et d'une paroi sonore. Suivant le rythme de l'envoi du courant électrique dans la bobine, celle-ci attire ou repousse l'aimant. Si elle le repousse, celui-ci met en mouvement la paroi sonore,

créant ainsi une vibration qui se transmet et s'ammplifie dans l'air, jusqu'à parvenir à votre oreille.

Le rythme de l'envoi du courant électrique permet également de déterminer la fréquence et donc la hauteur du son.

## 3.4 INSTRUCTIONS SONORES

Les principales instructions sonores du CPC sont:

SOUND

ENV

ENT

RELEASE

L'instruction SOUND est l'instruction de base pour produire des sons sur votre CPC. Elle a les paramètres suivants:

1. Etat canaux: cette valeur indique à l'ordinateur quel canal sonore doit être employé et quelles autres activités doivent être activées telles qu'un rendez-vous ou un arrêt par exemple.

Les valeurs de ce paramètre sont calculées d'après la signification suivante des différents bits:

| Bit | Valeur | Commande |
|---|---|---|
| 0 | 1 | son envoyé au canal A |
| 1 | 2 | son envoyé au canal B |
| 2 | 4 | son envoyé au canal C |
| 3 | 8 | rendez-vous avec canal A |
| 4 | 16 | rendez-vous avec canal B |
| 5 | 32 | rendez-vous avec canal C |
| 6 | 64 | arrêt |
| 7 | 128 | flush |

Si vous souhaitez par exemple envoyer un son sur tous les canaux,

le paramètre état canaux doit donc recevoir la valeur 7.

Le rendez-vous vous permet de synchroniser les trois canaux sonores et donc de faire exécuter une mélodie avec des pauses, sans que celà occasionne un craquement désagréable du haut-parleur.

Voici comment se présente un rendez-vous:

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Canal A: | NO | NO | NO | RB | RB |
| Canal B: | NO | RC | NO | NO | RA |
| Canal C: | NO | RB | RA | NO | NO |

NO signifie qu'une note est jouée, RB correspond à un rendez-vous avec le canal B.

Voici maintenant un schéma représentant le déroulement dans le temps de la structure de rendez-vous ci-dessus. Les actions synchronisées figurent dans la même colonne. NO correspond à une note, PA à une pause:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Canal A: | NO | NO | NO | PA | NO | NO | PA | PA |
| Canal B: | NO | NO | NO | NO | NO | NO | PA | PA |
| Canal C: | NO | NO | PA | PA | PA | NO | NO | NO |

Lorsqu'un rendez-vous est arrangé, une pause se produit dans le canal appelé. Le canal appelé est celui qui continue à jouer.

Une autre possibilité de synchroniser les canaux consiste à mettre

un arrêt. Un arrêt vous permet d'arrêter l'émission de son d'un canal déjà plein puis de la relancer avec l'instruction RELEASE. L'avantage par rapport au rendez-vous est que le canal sonore peut être rempli pendant un arrêt alors qu'un rendez-vous n'admet aucun son avant que le rendez-vous ne soit exécuté.

2. Valeur de la note: le second paramètre après l'instruction SOUND est la valeur de la note. Vous pouvez trouver les valeurs des différentes notes dans une annexe de votre manuel. Vous n'avez normalement besoin que des valeurs de l'octave 0 et vous pouvez ainsi calculer les autres valeurs:

Valeur = Valeur de la note de l'octave 0 / 2↑octave

Exemple: La valeur du DO de l'octave -2 sera: 478/2↑-2 ou 478/0.25

3. Durée du son: cette valeur doit être donnée en centièmes de seconde.

Si la valeur de ce paramètre est négative, celui-ci indique combien de fois la durée de ENV doit être répétée. Si cette valeur est nulle, le son sera joué conformément à la durée de ENV.

4. Volume: paramètre compris entre 0 et 7 lorsqu'il n'y a pas d'ENV et entre 0 et 15 s'il y a une ENV.

Les deux paramètres suivants seront expliqués plus tard, en liaison avec ENV et ENT.

Le dernier paramètre sert à la production de bruits. Si celle-ci

est utilisée sur les trois canaux, les trois canaux reçoivent le même bruit.

L'instruction suivante la plus étroitement liée à l'instruction SOUND est l'instruction RELEASE qui permet de supprimer un arrêt préalablement posé. Il suffit d'indiquer comme paramètre le numéro du canal (A=1, B=2, C=4).

Avant de passer maintenant à la partie la plus importante, l'explication des courbes d'enveloppe, nous vous proposons de mettre un peu d'ambiance en tapant le programme suivant qui joue "Oh when the saints go marching in". N'oubliez pas de le sauvegarder sur cassette car nous en aurons besoin plus tard.

```
10 READ a,b
20 IF a=-1 THEN RESTORE:GOTO 10
30 SOUND 1,a,b
40 SOUND 2,0.5*a,b
50 SOUND 4,0.25*a,b
60 GOTO 10
70 DATA 478,50,379,50,358,50,319,200,0,5,319,50,478,50,379,5
0,358,50,319,200,0,5,319,100
80 DATA 478,50,379,50,358,50,319,100,379,100,478,100,379,100
,426,200
90 DATA 0,5,426,50,379,50,0,5,379,50,426,50,478,150,0,5,478,
50
100 DATA 379,100,319,100,0,5,319,50,358,150,0,5,358,100,379,
50,358,50,319,100,379,100
110 DATA 478,100,426,100,478,200,0,5,478,50,-1,1
```

## 3.5 L'INSTRUCTION ENV ET LES COURBES D'ENVELOPPE DU VOLUME

L'instruction ENV modifie le volume d'un son de façon à imiter le jeu des "vrais" instruments de musique.

En augmentant ou en diminuant le volume d'un son pendant le jeu, on peut donc obtenir une meilleure sonorité.

En utilisant l'instruction ENV il faut en principe que le volume ait été mis à 0 avec l'instruction SOUND (sauf si vous voulez délibérément une valeur différente) pour que l'enveloppe fonctionne pleinement.

Les courbes d'enveloppe sont affectées à des symboles ou numéros d'enveloppe qui peuvent ensuite être employés dans l'instruction SOUND pour appeler une courbe d'enveloppe.

Ce symbole est le premier paramètre de l'instruction ENV.
Les trois paramètres suivants: nombre de pas, pas et temps de pause peuvent être employés 5 fois les uns à la suite des autres.

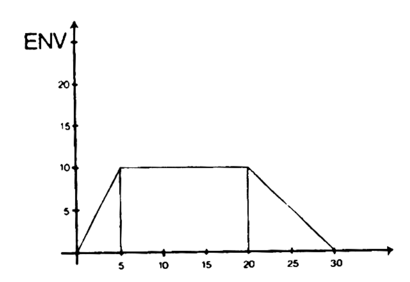
Le nombre de pas indique en combien de pas le volume doit augmenter ou diminuer. Le pas, multiplié par le temps de pause donne la durée du son. Si cette valeur est négative, le volume diminue. Le temps de pause est la durée en centièmes de seconde pendant laquelle doit être maintenu le volume actuel. Ce temps de pause peut être également appelé durée du pas.

Essayez maintenant avec notre petite chanson l'ENV suivante:

ENV 1,5,2,1,1,0,16,5,-3,2

Le résultat vous étonnera certainement mais nous allons maintenant vous expliquer comment nous avons calculé ces valeurs:

Votre manuel comprend une page sur laquelle vous pouvez dessiner les courbes de vos enveloppes. Pensez bien à noter les trois paramètres pour chaque partie de la courbe. Voici un dessin qui vous montre comment calculer les paramètres d'une ENV.



## 3.6 ENT ET LA COURBE DE L'ENVELOPPE DU SON

L'enveloppe du son est dans son principe analogue à l'enveloppe du volume mais les possibilités ne sont pas aussi grandes que pour l'enveloppe du volume. Les modifications de l'enveloppe du son permettent notamment de réaliser un vibrato c'est-à-dire une modification minime de la hauteur du son qui donne une impression de tremblement.

La structure de cette instruction et de ses paramètres de nombre de pas, de pas et de temps de pause est identique à celle d'ENV mais alors qu'ENV ne nécessite en principe que trois parties, attaque, soutien et relâche, ENT en nécessite le plus souvent 5.

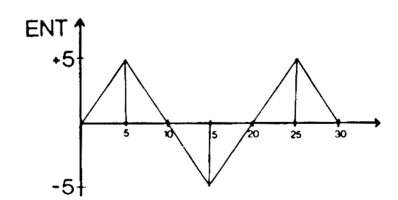
L'instruction suivante produit une sorte de vibrato:

```
ENT 1,5,1,1,10,-1,1,10,11,5,-1,1
```

N'oubliez pas dans le programme "Oh when the saints" d'appeler maintenant également ENT avec la valeur 1 qui correspond au numéro de notre ENT. Si vous employez une valeur négative, l'enveloppe est répétée.

Voici la courbe qui représente notre enveloppe. Nous avons choisi les mêmes unités que pour la courbe ENV.

Suivez le même schéma pour calculer les valeurs de vos enveloppes.




Il faut enfin remarquer que nos enveloppes ne sont pas de véritables courbes puisqu'il ne nous est pas possible d'indiquer les valeurs de tous les points d'une courbe mais seulement celles des angles de nos courbes. Il s'agit donc plutôt de vibrations carrées, se composant de segments de droites.

Ces deux instructions vous permettent cependant théoriquement de modifier chaque son jusqu'à le rendre méconnaissable, ce qui peut être très intéressant pour un synthétiseur. L'éditeur de son que nous vous fournissons plus loin sera à cet égard un excellent point de départ.

La méthode la plus simple pour bien comprendre le principe des enveloppes consiste à se livrer au plus grand nombre possible d'expériences car une sonorité voulue ne peut être obtenue uniquement par des calculs abstraits.

Pour simuler un instrument, vous devez savoir comment il sonne exactement: un instrument à corde émet le plus souvent un petit écho dû à la vibration des cordes. Pour imiter ce phénomène, il faut donc que la troisième phase d'ENV, la tombée du volume soit

plus longue que la première phase d'attaque du son.
Pour les instruments à percussion tels que batterie ou timbale, cet écho ne se produit pas et le son doit donc retomber plus vite qu'il n'est monté.

Mais il est nettement plus compliqué d'imiter un instrument à vent. Il y a en effet différentes façon de jouer de la trompette ou du trombone. Le plus important est de faire monter le volume de manière brutale (plus ou moins suivant la taille du corps de résonance) et de faire dépendre la retombée du volume du type de jeu qu'on veut imiter.

## 3.7 L'EDITEUR MUSICAL

Comme vous l'avez certainement constaté, l'emploi des instructions musicales est tout de même relativement compliqué puisque vous êtes obligé en permanence de faire des calculs de paramètres. C'est pourquoi nous vous proposons un éditeur musical qui vous permettra de produire des sons simplement en appuyant sur une touche du clavier. L'affectation des touches correspond à la gamme et va donc de do à si et les demi-tons (les dièzes) peuvent être obtenus avec la touche SHIFT.

Le programme vous permet de:

1. programmer des mélodies
2. sauvegarder des mélodies
3. faire jouer des mélodies
4. charger des mélodies

1. Ce premier point du programme vous permet de produire des sons en appuyant sur différentes touches du clavier. Ces sons sont mis en mémoire par votre CPC qui pourra les rejouer lorsque vous le souhaiterez. Vous disposez de 9 octaves que vous pouvez sélectionner en appuyant sur une touche entre 1 et 9. L'affectation des touches sera récapitulée plus loin dans un tableau. En appuyant sur la touche 'P' vous pouvez faire afficher à l'écran la dernière touche sur laquelle vous avez appuyé.
Vous devez cependant faire attention à la limite suivante: vous ne pouvez faire mettre en mémoire plus de 1000 sons pour ne pas épuiser la capacité de mémoire disponible et pour éviter des complications lors de la sauvegarde. Surveillez donc le nombre de

sons déjà mis en mémoire qui s'affiche dans l'angle supérieur gauche de l'écran.

Vous pouvez définir la durée d'un son en tenant une touche appuyée plus ou moins longtemps. Il se produit toutefois une petite pause lors de la programmation qui disparaîtra lors de l'exécution de la mélodie.

Pour revenir au menu lorsque vous êtes dans le mode 1 (programmation de mélodies), appuyez sur la touche ENTER.

2. Vous pouvez ici sauvegarder une mélodie en en indiquant le nom. Les notes sont sauvegardées sous la forme d'une liste de variables que le mode 4 vous permet de recharger. Laissez toujours un certain espace entre deux mélodies différentes pour pas qu'elles ne se chevauchent et pour que vous puissiez les retrouver plus facilement.

3. Ce mode vous permet de faire jouer la mélodie que vous venez de programmer ou de faire charger. Les notes sont jouées dans l'ordre où vous les avez entrées, sans pause. Pour programmer une pause, il faut appuyer en mode 1, lors de la programmation de la mélodie, sur la touche espace.

4. Ce mode vous permet de faire charger une mélodie que vous pouvez ensuite faire exécuter en mode 3. Il n'est cependant pas possible de modifier une mélodie que vous venez de charger. Pour

que ceci soit possible, il faudrait que vous modifiez légèrement le programme. Vous devriez y arriver sans trop de difficultés une fois que vous aurez étudié le commentaire du listing de ce programme.

Autre points susceptibles d'être améliorés:

a) le programme ne travaille qu'avec un canal, ce à quoi vous pouvez remédier en ajoutant deux instructions SOUND et en étendant l'affectation des touches.

b) vous n'avez aucune possibilité de modifier le volume en mode programme mais vous pouvez le faire avec le bouton de volume de votre CPC ou de votre chaîne stéréo.

Pour utiliser pleinement notre programme il est recommandé de connecter le CPC à une chaîne stéréo car la sonorité ne sera pas sinon très bonne.

Table d'affectation des touches:

--------------------------------

| Touche | Signification | Commentaire |
|---|---|---|
| c | note do | |
| C | do dièze | |
| d | note ré | |
| D | ré dièze | |
| e | note mi | |
| f | note fa | |
| F | fa dièze | |
| g | note sol | |

| | | |
|---|---|---|
| G | sol dièze | |
| a | note la | |
| A | la dièze | |
| b | note si | |
| espace | pause | |
| p | affichage touches | activé la première fois, désactivé la seconde |
| 1 | octave -3 | numéro d'octave d'après la table du manuel |
| 2 | octave -2 | |
| 3 | octave -1 | |
| 4 | octave 0 | |
| 5 | octave 1 | |
| 6 | octave 2 | |
| 7 | octave 3 | |
| 8 | octave 4 | |
| 9 | octave 5 | non indiquée dans le manuel |
| ENTER | fin | retour au menu |

```
10 DIM ton(1000)
20 MODE 2
30 LOCATE 5,10:PRINT"Programmer une melodie    :1"
40 LOCATE 5,12:PRINT"Sauvegarder une melodie   :2"
50 LOCATE 5,14:PRINT"Jouer une melodie         :3"
60 LOCATE 5,16:PRINT"Charger une melodie       :4"
70 LOCATE 5,20:INPUT"Votre choix";wa
80 ON wa GOTO 100,490,330,390
90 END
100 CLS
110 PRINT"P r o g r a m m e r   u n e   m e l o d i e"
120 n=0
130 FOR toene=1 TO 1000
140 a$=INKEY$:IF a$="" THEN 140
150 a=a+(478 AND a$="c")+(426 AND a$="d")
160 a=a+(379 AND a$="e")+(358 AND a$="f")
170 a=a+(319 AND a$="g")+(284 AND a$="a")
180 a=a+(253 AND a$="b")+(451 AND a$="C")
190 a=a+(402 AND a$="D")+(338 AND a$="F")
200 a=a+(301 AND a$="G")+(268 AND a$="A")
210 IF a$="p" AND pr=0 THEN pr=1:GOTO 140
220 IF a$="p" AND pr=1 THEN pr=0
230 IF a$=CHR$(13) THEN GOTO 20
240 nn=VAL(a$):IF nn=0 THEN n=n ELSE goto320
250 IF pr=1 THEN PRINT a$;
260 ton(toene)=(a*2^n)
270 SOUND 1,ton(toene)
280 LOCATE 1,1:PRINT toene
290 a=0
300 NEXT
310 GOTO 20
320 n=4-nn:GOTO 140
330 CLS
340 LOCATE 5,15:PRINT"Ecoutez la melodie"
350 FOR n=1 TO toene
360 SOUND 1,ton(n)
370 NEXT
380 GOTO 20
390 CLS
400 LOCATE 5,5:PRINT"C h a r g e r   u n e   m e l o d i e"
410 LOCATE 1,20
415 INPUT"Nom de la melodie";na$
420 OPENIN na$
430 INPUT£9,toene
440 FOR n=1 TO toene
450 INPUT£9,ton(N)
460 NEXT N
470 CLOSEIN
480 GOTO 20
490 CLS
500 LOCATE 5,5:PRINT"S a u v e g a r d e r   u n e   m e l o d i e"
510 INPUT"Nom de la melodie";na$
520 OPENOUT na$
530 PRINT£9,toene
540 FOR n=1 TO toene
550 PRINT£9,ton(n)
560 NEXT
570 CLOSEOUT
580 GOTO 20
```

**Commentaire du listing:**

---

| | |
|---|---|
| lignes 20-90 | :menu et choix du mode (1-4) |
| lignes 100-310 | :mode programmation |
| ligne 130 | :ouverture de la boucle pour le nombre de notes |
| ligne 140 | :interrogation du clavier |
| lignes 150-230 | :évaluation de la touche enfoncée |
| ligne 240 | :évaluer si octave a été changée |
| ligne 250 | :affichage éventuel de la touche enfoncée |
| ligne 260 | :écriture de la valeur dans la variable actuelle du tableau |
| ligne 270 | :production du son par l'instruction SOUND |
| ligne 310 | :retour au menu |
| ligne 320 | :changement de l'octave, retour au mode programmation |
| lignes 330-380 | :mode de jeu |
| ligne 350 | :ouverture de la boucle pour le nombre de sons |
| ligne 360 | :jeu de la note actuelle |
| ligne 380 | :retour au menu |
| lignes 390-480 | :charger une mélodie |
| ligne 420 | :ouverture d'un fichier d'entrée cassette |
| ligne 430 | :lecture du nombre de sons |
| ligne 440 | :ouverture d'une boucle de lecture des sons |
| ligne 450 | :lecture des sons |
| ligne 470 | :fermeture du fichier d'entrée cassette |
| ligne 480 | :retour au programme principal |
| lignes 490-580 | :sauvegarde d'une mélodie |
| ligne 510 | :entrée du nom de la mélodie |
| ligne 520 | :ouverture d'un fichier sortie cassette |

| | |
|---|---|
| ligne 530 | :écriture du nombre de sons sur la bande |
| ligne 540 | :ouverture de la boucle pour l'écriture des sons sur la bande |
| ligne 550 | :écriture des différents sons sur la bande |
| ligne 570 | :fermeture du fichier sortie cassette |
| ligne 580 | :retour au programme principal |

## 3.8 PROGRAMMES EXEMPLES

Les petits programmes suivants vous permettront d'ajouter à vos propres programmes les sons qui leur manquent. Nous avons choisi des exemples que vous pourrez intégrer sans difficultés dans nos deux jeux. Il vous sera également aisé de les modifier suivant vos besoins.

Les titres de chaque programme correspondent à ce que nous avons entendu. Vous pouvez bien sûr les utiliser pour imiter d'autres sons.

Une autre remarque: les numéros de ligne peuvent fort bien être modifiés, à condition bien sûr que vous pensiez à modifier également les numéros de ligne des instructions de saut, ce qui est très facile avec l'instruction RENUM dont dispose votre ordinateur.

Vous pouvez enfin améliorer nos exemples en utilisant les enveloppes ou créer des dessins correspondant aux sons de nos exemples. Plus vous donnerez cours à votre imagination, meilleurs seront les résultats obtenus.

```
10 REM Police
20 FOR n=100 TO 200 STEP 10
30 SOUND 1,n,2
40 NEXT
50 FOR n=200 TO 100 STEP -10
60 SOUND 1,n,2
70 NEXT
80 GOTO 20
```

```
10 REM Occupé
20 SOUND 1,100,100,15
30 SOUND 1,0,100
40 GOTO 20
```

```
10 REM Bataille galaxiale
20 FOR n=90 TO 125 STEP INT(RND(1)*10)+1
30 SOUND 1,n,2,15
40 NEXT
50 SOUND 1,0,INT(RND(1)*20)
60 GOTO 20
```

```
10  REM Gamme
20 FOR n=127 TO 450  STEP 12
30 SOUND 1,n
40 NEXT n
50 FOR n=450 TO 127  STEP -12
60 SOUND 1,n
70 NEXT n
80 GOTO 20
```

```
10 REM Explosion
20 FOR n=15 TO 1 STEP -1
30 SOUND 1,426,40,n,,,1
40 NEXT
```

```
10 REM Alarme
20 FOR n=500 TO 100 STEP -15
30 SOUND 1,n,4
40 NEXT
50 SOUND 1,0,30
60 GOTO 20
```

```
10 REM Touches lointaines
20 FOR n=100 TO 800 STEP 15
30 SOUND 1,n,2,15
40 NEXT
50 FOR x=1 TO 7 STEP 0.5
60 SOUND 1,400,5,x,,,1
70 NEXT
80 SOUND 1,0,50
90 GOTO 20
```

```
10 REM Mélodie favorite
20 a=INT(RND(1)*3500)+284
30 SOUND 1,a
40 b=INT(RND(1)*3400)+284
50 SOUND 2,b
60 c=INT(RND(1)*3300)+284
70 SOUND 4,c
80 GOTO 20
```

Le programme suivant répond au même principe que le programme musical "Oh when the saints". Ce programme joue la mélodie "Happy Birthday". Comme il ne comprend que des notes et des durées de notes, nous vous conseillons également de l'améliorer avec ENV et ENT pour obtenir une sonorité plus belle.

```
10 REM happy birthday
20 READ a,b
30 IF b=-1 THEN END
40 SOUND 1,a,b
50 GOTO 20
60 DATA 319,50,319,50,284,100,319,100,239,100
70 DATA 253,150,319,50,319,50,284,100,319,100
80 DATA 213,100,239,150,319,50,319,50,159,100
90 DATA 190,100,239,100,253,100,284,100,179,50
100 DATA 179,50,190,100,239,100,213,100,239,150
110 DATA 0,-1
```

Commentaire du listing:

----------------------

10 titre

20 lecture des notes et durées de notes

30 demande si morceau terminé

40 joue la note A de durée B

50 retour en 20

60-110 data de notes et durées de notes

### 3.8.1 REVEIL SONORE

Pour vous distraire de nos explications parfois très théoriques sur la création du son avec le CPC 464, voici un programme que vous pourrez fort bien utiliser si vous faites partie de ceux qui ont un peu de mal à se lever le matin. Il s'agit d'une horloge digitale qui est conçue de façon à pouvoir vous réveiller à coup sûr. Elle émet un léger son pour marquer chaque seconde, un son plus fort pour marquer chaque minute et comme elle joue une petite mélodie pour marquer les heures, vos amis penseront certainement que vous avez une véritable horloge suisse.

Notre horloge est encore simple mais vous pouvez en faire un second big ben si vous calculez les notes de morceaux de musique que vous souhaitez faire jouer par le réveil. Vous pouvez également vous faire réveiller par la mélodie ou par un son encore plus terrible que celui que nous avons choisi.

Mode d'emploi:
--------------

Après que vous ayez lancé le programme, on vous demande d'entrer l'heure actuelle dans l'ordre heures,minutes,secondes.
Entrez ces valeurs l'une après l'autre, avec la touche ENTER.
On vous demande alors l'heure du réveil. L'entrée se fait de la même façon. Si vous appuyez sur ENTER sans avoir tapé de chiffre, c'est à minuit (0 heure) que vous serez réveillé.
Pour interrompre le signal du réveil, appuyez sur une touche. Le signal du réveil s'arrêtera alors et l'horloge continuera à fonctionner normalement.

Si vous constatez que l'horloge n'est pas exacte, modifiez en ligne 150 le pas de l'instruction EVERY. Si l'horloge retarde, il faut que vous diminuiez la valeur suivant EVERY et que vous l'augmentiez au contraire si l'horloge avance.

```
10 MODE 2
20 INPUT "heures   ";z
30 INPUT "minutes  ";y
40 INPUT "secondes ";x
50 INPUT "heure de réveil (h)",z2
60 INPUT "heure de réveil (m)",y2
70 INPUT "heure de réveil (s)",x2
80 MODE 0
90 LOCATE 3,9:PRINT CHR$(150);STRING$(&D,CHR$(154));CHR$(156
)
100 LOCATE 3,10:PRINT CHR$(149):LOCATE 17,10:PRINT CHR$(149)

110 LOCATE 3,11:PRINT CHR$(149):LOCATE 17,11:PRINT CHR$(149)

120 LOCATE 3,12:PRINT CHR$(149):LOCATE 17,12:PRINT CHR$(149)
130 LOCATE 3,13:PRINT CHR$(147);STRING$(&D,CHR$(154));CHR$(1
53)
140 LOCATE 5,5:PRINT" R E V E I L "
150 EVERY 50,2 GOSUB 170
160 GOTO 160
170 SOUND 1,284,5,5:x=x+1
180 IF x=60 THEN x=0:y=y+1:SOUND 2,71,5,6
190 IF y=60 THEN y=0:z=z+1:GOSUB 270
200 IF z=z2 AND x=x2 AND y=y2 THEN GOSUB 320
210 LOCATE 5,11:PRINT z
220 LOCATE 8,11:PRINT":"
230 LOCATE 9,11:PRINT y
240 LOCATE 12,11:PRINT":"
250 LOCATE 13,11:PRINT x
260 RETURN
270 READ no
280 IF no=-1 THEN RETURN
290 SOUND 4,no,30,7
300 GOTO 270
```

```
310 DATA 253,253,169,169,150,150,169,190,190,201,201,225,225
,253, 169,169,190,190,201,201,225,169,169,190,201,201
315 DATA 225,253,253,169,169,150,150,169,190,190,201,201,225
,225,223,-1
320 SOUND 7,100,1000,15
330 IF INKEY$="" THEN GOTO 330
340 SOUND 135,0
350 RETURN
```

Commentaire du listing:

-----------------------

| | |
|---|---|
| lignes 10-80 | fixation du mode écran, entrée des valeurs nécessaires (h,m,s) pour l'heure et l'heure de réveil |
| lignes 90-140 | dessin du cadre de l'horloge |
| ligne 150 | l'instruction EVERY appelle à chaque seconde la routine 170 |
| ligne 170 | production du signal des secondes et comptage d'une seconde |
| ligne 180 | interroge si minute atteinte; si oui, production du signal des minutes |
| ligne 190 | interroge si heure atteinte; si oui, production du signal des heures |
| ligne 200 | interroge si heure du réveil atteinte; si oui, saut à la sous-routine en 320 |
| lignes 210-260 | affichage de l'heure dans le cadre |
| ligne 270 | lecture des notes |
| ligne 280 | interroge si fin de la mélodie atteinte |
| ligne 290 | joue la mélodie avec les trois canaux sonores |
| ligne 310 | ligne de data avec les valeurs de notes |
| ligne 320 | production du signal du réveil |
| ligne 330 | le programme attend ici qu'on appuie sur une touche |
| ligne 340 | ce type d'instruction SOUND interrompt le signal |

du réveil et envoie un arrêt dans les canaux sonores.

# CHAPITRE 4 LE LANGAGE MACHINE

## 4.1 INTRODUCTION AU LANGAGE MACHINE

Cette introduction n'est pas un cours exhaustif sur le langage-machine. Elle doit simplement vous permettre d'appréhender les techniques de programmation les plus sophistiquées que vous puissiez appliquer à votre CPC 464. Vous savez en effet certainement que certains problèmes de programmation, insolubles en BASIC, ne peuvent être résolus qu'au moyen de petites routines en langage-machine. Une routine de tri beaucoup trop longue en BASIC pourra, écrite en BASIC, prendre jusqu'à 100 fois moins de temps.

Le cerveau de votre CPC 464 est un microprocesseur Z 80, un des microprocesseurs 8 bits les plus répandus dans le domaine des ordinateurs familiaux. Le Z 80 possède un jeu d'instructions de plus de 600 instructions dont nous ne pouvons vous donner une description complète dans le présent ouvrage.

Le microprocesseur de votre CPC 464 travaille de manière digitale, c'est-à-dire qu'il ne peut, dans toute opération interne, distinguer que deux états, que nous pouvons nous représenter comme les états ALLUME ou ETEINT ou bien 1 ou 0. Un tel commutateur, qui ne peut donc représenter que deux états, ne permet pas d'aller bien loin. C'est pour celà que huit commutateurs sont toujours regroupés, à l'intérieur du microprocesseur, pour former un registre. On désigne un commutateur sous le nom de bit, alors qu'on appelle octet un groupe de huit bits. Un registre de 8 bits peut être ainsi représenté:

7 6 5 4 3 2 1 0

0 1 1 0 1 0 0 1

A la ligne supérieure figure la numérotation des bits, qui va de 0 à 7. En dessous, le contenu de chaque bit, soit 0, soit 1. Alors qu'un bit ne peut donc représenter que deux états, 8 bits peuvent déjà en représenter plus. Si nous comparons avec les nombres décimaux auquels nous sommes habitués,

3 2 1 0

5 7 2 4

nous avons, ici aussi, numéroté chaque chiffre, cette fois de 0 à 3. Comment calculons-nous la valeur d'un tel nombre? Chaque chiffre a une valeur entre 0 et 9 qui représente une grandeur 10 fois plus importante chaque fois qu'on se déplace d'un chiffre vers la gauche:

4 + 2*10 + 7*10*10 + 5*10*10*10 = 5724

Nous pouvons procéder, avec le contenu de notre registre, de manière tout à fait analogue: on appelle de tels nombres, par opposition aux nombres décimaux, nombres binaires puisqu'un chiffre peut prendre, non pas 10, mais seulement 2 valeurs. Un chiffre plus à gauche ne représente donc plus non plus une valeur 10 fois mais seulement 2 fois supérieure. Nous pouvons maintenant calculer également la valeur du contenu de notre registre:

1 + 0*2 + 0*2*2 + 1*2*2*2 + 0*2*2*2*2

1*2*2*2*2*2 + 1*2*2*2*2*2*2 + 0*2*2*2*2*2*2*2

= 1 + 0 + 0 + 8 + 0 + 32 + 64 + 0 = 105

Nous obtenons donc, comme contenu du registre, la valeur 105. Nous pouvons nous simplifier ce calcul laborieux en calculant d'abord une bonne fois les valeurs de chaque chiffre, de même que nous avons en tête les valeurs des chiffres des nombres décimaux.

| Chiffre | Valeur |
|---|---|
| 0 | 2↑0 = 1 |
| 1 | 2↑1 = 2 |
| 2 | 2↑2 = 4 |
| 3 | 2↑3 = 8 |
| 4 | 2↑4 = 16 |
| 5 | 2↑5 = 32 |
| 6 | 2↑6 = 64 |
| 7 | 2↑7 = 128 |

Quelle peut être la valeur maximale du contenu d'un registre? La valeur maximale est atteinte lorsque tous les chiffres ont la valeur 1, ce qui donne alors: 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 = 255. La valeur la plus élevée qui puisse être représentée avec 8 bits est donc 255. On peut ainsi représenter 2↑8 = 256 valeurs différentes (de 0 à 255).

Pour convertir les nombres binaires en décimaux il suffit d'entrer PRINT&xnombre binaire. Pour convertir les nombres décimaux en binaires il suffit d'entrer PRINT BIN$(nombre décimal):

```
PRINT&x01010101
85
```

```
PRINTBIN$(85)
01010101
```

Notez que les 0 placés avant le 1 le plus à gauche sont ignorés.

## 4.2 LE SYSTEME HEXADECIMAL

La manipulation des nombres binaires reste toutefois très fastidieuse. C'est pourquoi a été introduit un deuxième système numérique, qui reste cependant en relation étroite avec les nombres binaires, mais qui permet une représentation des valeurs avec beaucoup moins de chiffres. Si l'on divise, en effet, un nombre de 8 bits en 2 nombres de 4 bits, chacun des 2 nombres de 4 bits pourra représenter 16 valeurs différentes. Il suffit alors de créer un système numérique possédant 16 chiffres différents pour pouvoir exprimer, avec 2 chiffres de ce système, tout nombre binaire de 8 chiffres.

7 6 5 4 3 2 1 0

0 1 1 0 1 0 0 1

6 9

Chaque octet est ainsi divisé en 2 demi-octets, également appelés quartets. Cependant, comme chaque quartet peut prendre une valeur allant de 0 à 15 et que notre système décimal ne comporte que des chiffres allant de 0 à 9, se pose le problème des chiffres devant représenter les valeurs 10 à 15. Ce problème a été résolu en prenant simplement les lettres A à F. Le système numérique ainsi réalisé a été appelé "hexadécimal". Voici un tableau d'équivalence des trois systèmes évoqués:

| Binaire | Hexadécimal | Décimal |
|---|---|---|
| 0000 | 0 | 0 |
| 0001 | 1 | 1 |
| 0010 | 2 | 2 |
| 0011 | 3 | 3 |
| 0100 | 4 | 4 |
| 0101 | 5 | 5 |
| 0110 | 6 | 6 |
| 0111 | 7 | 7 |
| 1000 | 8 | 8 |
| 1001 | 9 | 9 |
| 1010 | A | 10 |
| 1011 | B | 11 |
| 1100 | C | 12 |
| 1101 | D | 13 |
| 1110 | E | 14 |
| 1111 | F | 15 |

Nous utiliserons essentiellement les nombres hexadécimaux parce qu'ils sont d'une part faciles à représenter et qu'ils sont aisément traduisibles en nombres binaires. Le système hexadécimal permet d'exprimer avec seulement deux chiffres les valeurs de tout groupe de 8 bits.

Les nombres de deux octets (donc de 16 bits) sont divisés en deux: l'octet fort et l'octet faible (highbyte et lowbyte). Dans le nombre A1E1, A1 sera l'octet fort (highbyte) et E1 l'octet faible (lowbyte). Pour calculer un nombre de 16 bits il suffit donc de calculer les valeurs de ses octets fort et faible puis de

multiplier la valeur obtenue pour l'octet fort par 256 et de l'ajouter à la valeur obtenue pour l'octet faible:

A1E1 => octet fort = A1 = 161
        octet faible = E1 = 225
A1E1 = 161*256+225 = 41441

La formule est: octet fort * 256 + octet faible

Bien sûr vous pouvez vous dispenser de faire ces calculs de conversion puisque votre CPC 464 dispose également de fonctions de conversion décimal-hexadécimal:

PRINT HEX$(nombre décimal)

et hexadécimal-décimal:

PRINT&nombre hexadécimal

Il y a toutefois une limite: cette fonction ne permet de convertir que les nombres hexadécimaux jusqu'à 7FFF car pour les nombres plus importants, 65536 est retranché du résultat ce qui donne un nombre négatif:

PRINT&A1E1 ne donne pas 41441 mais -24095 (41441-65536=-24095).

Pour remédier à cet inconvénient, vous pouvez utiliser la routine suivante:

```
10INPUT A$:B$="&"+A$:IF VAL(B$)<0 THEN PRINT VAL(B$)+65536:END
```

```
20 PRINT VAL(B$)
```

Nous avons utilisé une variable alphanumérique pour que vous ne soyez pas toujours contraint d'écrire un "&".

Imprégnez-vous bien de ces notions fondamentales d'octet fort et d'octet faible car nous les retrouverons très souvent dans la suite de cet ouvrage, notamment pour le calcul des adresses. C'est le même principe qui explique d'ailleurs que l'adresse la plus élevée soit 65535 puisque l'octet fort peut au maximum contenir 255 ainsi que l'octet faible. Une adresse sur deux octets peut donc au maximum valoir 255*256+255 = 65535.

## 4.3 TECHNIQUES DE PROGRAMMATION

Plusieurs méthodes permettent de programmer en langage-machine à partir du BASIC: la première et la plus longue en temps consiste à POKEr les différents codes d'instruction dans les adresses voulues. Une méthode beaucoup plus pratique consiste à utiliser ce qu'on appelle un moniteur de langage-machine. Nous vous fournissons à la fin de ce chapitre un moniteur provisoire de langage-machine. Ce programme sera décrit en détail plus loin.

En ce qui concerne la première méthode: vous savez que l'instruction POKE X,A écrit dans l'adresse X la valeur A. En langage-machine les instructions sont affectées à des adresses de la mémoire, comme elles le sont en BASIC à des numéros de ligne. Si vous voulez donc écrire un programme en langage-machine à partir de l'adresse &2000, il faut poker la valeur correspondant à la première instruction du programme dans l'adresse &2000 et les valeurs correspondant aux paramètres, s'il y a lieu, dans les adresses &2001 et &2002. Si la première instruction tient sur un octet, l'adresse &2001 contiendra déjà la seconde instruction.

Les programmes en langage-machine ne se composent que de codes directement compréhensibles par l'ordinateur. A chaque instruction correspond en effet un ou plusieurs codes, ce qui fait qu'un programme en langage-machine n'est qu'une longue suite de chiffres qu'il faut entrer en mémoire avec l'instruction POKE dans l'ordre correspondant à la suite logique des instructions.

Pour lire le contenu d'une adresse de la mémoire de votre ordinateur, vous devez utiliser l'instruction PEEK(X). Le résultat

obtenu est le contenu de l'adresse X. Pour lancer un programme déjà placé en mémoire, il faut utiliser l'instruction CALL X qui appelle le programme en langage-machine figurant à l'adresse X.

Vous allez voir surgir dans les pages suivantes de nombreux problèmes qui ne doivent pas vous impressionner: nous n'avons pas nous non plus appris le langage-machine en trois jours.

## ASSEMBLEUR

L'assembleur est un langage de programmation qui utilise des abréviations symboliques à la place des codes véritables des instructions du langage-machine. Ces abréviations sont appelées "mnémoniques". Par exemple, l'instruction "load" est figurée par l'abréviation "ld". Comme la machine ne comprend que les codes binaires, il faut qu'un programme convertisse les mnémoniques d'instructions en nombres binaires. Un tel programme s'appelle un assembleur. Un assembleur permet en outre d'affecter aux adresses importantes des noms symboliques appelés labels (étiquettes). Les labels peuvent servir également à donner un nom aux variables que nous utilisons.

Comme l'indique le manuel, la société Amstrad prépare un programme d'assembleur.

## REGISTRES ET PREMIERES INSTRUCTIONS

Examinons tout d'abord la structure des registres du

microprocesseur (cerveau de l'ordinateur):

-le registre a, souvent appelé également accumulateur. Il sert essentiellement au stockage et au traitement des valeurs actuellement utilisées car il peut être employé de manière très polyvalente.

-le registre bc, ce qui signifie compteur d'octets (byte counter) sert souvent aux tâches de comptage. Il s'agit en fait d'un double registre puisqu'on peut appeler séparément les registres b et c qui le composent. Le registre bc, contrairement au registre a, peut donc admettre des nombres de deux octets (jusqu'à 65535).

-le registre hl (high-low) est également un registre double qui sert essentiellement à stocker des adresses qui, comme vous le savez, sont des nombres 16 bits. Toutefois, il est également possible d'appeler séparément les deux registres dont il se compose.

-le registre de est un autre registre double important. Ce registre permet essentiellement de stocker des valeurs 16 bits, soit quand les autres registres doubles sont déjà occupés, soit tout simplement lorsqu'une instruction a besoin de trois double-registres.

Lorsqu'on considère la structure du microprocesseur, on se demande tout d'abord comment une valeur déterminée peut être transférée de la mémoire dans un registre ou d'un registre dans un autre. Ceci est rendu possible par une instruction fondamentale, l'instruction de chargement ld (load):

ld a,n  charge la valeur n dans l'accumulateur. n est une entrée directe d'un octet. nn signifierait, bien sûr uniquement avec un registre double, une entrée immédiate également mais de deux octets (fort et faible). Si les deux nn sont entre parenthèses, celà signifie par contre qu'il ne s'agit plus d'une entrée directe mais que le contenu de l'adresse nn doit être chargé dans le registre considéré.

ld hl,(&A000) signifie que le contenu de l'adresse A000 doit être chargé dans le double-registre hl.

Pour charger le contenu de l'accumulateur dans un autre registre, par exemple le registre c, il suffit d'entrer: ld c,a. Le contenu antérieur du registre c sera effacé mais pas celui du registre a qui se retrouvera en outre dans le registre c.

Pour programmer en langage-machine, on a besoin d'une zone de stockage protégée, qui ne puisse pas être effacée par un programme BASIC. La séquence d'instructions suivante vous permet de constituer au dessus de la mémoire BASIC une zone libre dans laquelle vous pourrez placer les routines que nous allons décrire dans les pages suivantes:

```
MEMORY &1FFF:PRINT HIMEM:NEW
```

Si vous avez auparavant créé des caractères redéfinis, éteignez puis rallumez votre ordinateur pour les effacer car sinon vous pourriez avoir de petits problèmes.

Vous avez maintenant une zone libre de 34 kilooctets pour vos

programmes en langage-machine, mais il ne vous reste plus que 7.5 kilo octets pour vos programmes en BASIC.

Entrez maintenant:

POKE &2000,&3E:POKE &2001,&FF:POKE &2002,&C9

Les instructions POKE ci-dessus écrivent des instructions en langage-machine dans la mémoire.

Les valeurs placées à la suite de l'instruction POKE sont les adresses dans lesquelles doivent être placées les instructions. Elles sont comparables aux numéros de ligne en BASIC. La valeur &3E est le code de l'instruction ld a,n. La valeur &FF est l'opérande n de cette instruction. La valeur &C9 est mise pour RET (RETURN) qui marque à la fin d'un programme en langage-machine l'ordre de retourner au BASIC.

Lancez maintenant le programme avec CALL (&2000)

Après un très court instant, l'ordinateur affiche READY et vous ne pouvez encore vous rendre compte de ce qui s'est passé. Pourtant la valeur &FF a bien été chargée dans l'accumulateur, mais ceci n'a pas d'effet direct sur l'ordinateur.

Essayez maintenant d'entrer la routine: ld bc,&A1F1 en entrant:

POKE &2000,&1 (<=> ld bc,nn) :POKE &2001,&F1 (octet faible du nombre) :POKE &2002,&A1 (octet fort du nombre) : POKE &2003,&C9 (RET).

Lancez maintenant le programme avec CALL (&2000)

Après un très court instant, l'ordinateur affiche READY et vous ne

pouvez encore vous rendre compte de ce qui s'est passé. Pourtant la valeur &FF a bien été chargée dans l'accumulateur, mais ceci n'a pas d'effet direct sur l'ordinateur.

Essayez maintenant d'entrer la routine: ld bc,&A1F1 en entrant: POKE &2000,&1 (<=> ld bc,nn) :POKE &2001,&F1 (octet faible du nombre) :POKE &2002,&A1 (octet fort du nombre) : POKE &2003,&C9 (RET).

Vous ne voyez toujours aucune différence, mais la valeur &A1F1 a bien été chargée dans le double-registre bc. Notez à ce propos une règle fondamentale de la programmation qui est que tout opérande 16 bits d'une instruction doit toujours être présenté dans le format octet faible-octet fort (low-high).

Essayons maintenant des routines plus utiles, par exemple un programme d'addition de nombres de 1 octet.

Nous nommerons les deux octets à additionner ADD1 et ADD2 et nous les placerons respectivement dans les adresses &3001 et &3002; le résultat ira en &3004. Il nous faut maintenant trouver une possibilité d'additionner ces deux valeurs. Nous allons placer ADD1 dans l'accumulateur, et l'adresse d'ADD2 dans le registre hl: ld a,(&3000) et ld hl,&3002. Nous n'avons pas placé ADD2 directement en hl pour vous montrer les possibilités de la programmation.

L'instruction d'addition est: add a,(hl). Le contenu de l'adresse se trouvant dans le registre entre parenthèses (hl) sera donc additionné à l'accumulateur.

Pour placer le résultat, qui se trouve dans l'accumulateur, en &3004, nous avons l'instruction: ld (&3004),a. L'instruction RET nous permettra enfin de revenir au BASIC. Voici notre petit programme récapitulé avec les codes correspondants qu'il vous faudra entrer en mémoire:

```
ld a,(&3000)          &3A &00 &30
ld hl,&3002           &21 &02 &30
add a,(hl)            &86
ld (&3004),a          &32 &04 &30
RET                   &C9
```

Voici maintenant un petit programme qui vous permettra d'entrer ce programme à partir du BASIC:

```
10 FOR N=&2000 TO &200A
20 INPUT VALEUR
30 POKE N,VALEUR
40 NEXT
```

Il vous faut encore placer deux nombres (inférieurs à 128) dans les adresses &3000 et &3001: POKE &3000,ADD1:POKE &3001,ADD2.

Entrez CALL &2000. Vous pouvez maintenant vérifier avec ?PEEK(&3004) que le résultat de l'opération a bien été placé en &3004.

Examinons maintenant le fonctionnement de l'addition de codes binaires:

ADD1 vaut 127 et ADD2 64.

ADD1= 01111111

ADD2= 01000000

RES = 10111111 = 191 en décimal

Le principe est simple: si un bit d'une des deux valeurs à additionner est mis alors que le même bit de l'autre valeur n'est pas mis, le bit correspondant du résultat sera mis. Si les mêmes bits des deux valeurs ne sont pas mis, le bit correspondant du résultat vaudra également 0. Si les mêmes bits des deux valeurs à additionner sont mis, le bit correspondant du résultat vaudra 0 mais une retenue se produira qui sera prise en compte pour l'addition du bit placé à gauche du bit actuel (voyez dans notre exemple les bits 6 et 7). Si la retenue se produit lors du calcul du bit 7, bit le plus à gauche, celle-ci ne sera pas prise en compte dans les opérations sur 8 bits mais elle le sera par contre pour les opérations sur 16 bits. C'est pourquoi dans l'exemple ci-dessus nous vous demandions de ne pas utiliser de nombres supérieurs à 127, afin d'éviter qu'une retenue puisse se produire.

Essayons maintenant de réaliser une addition sur 16 bits. Il nous faudra tout d'abord deux adresses pour placer les valeurs à additionner: ADD1 ira en &3001 (ADR1) et &3000 (ADR1-1) et ADD2 ira en &3003 (ADR2) et &3002 (ADR2-1). Le résultat ira en &3005 (RES) et &3004 (RES-1).

Notre programme additionnera tout d'abord les deux octets faibles de nos deux valeurs. Si une retenue se produit, le flag carry

(signe de retenue) sera mis. Il sera exploité dans la seconde partie de notre programme.

La seconde partie du programme additionnera les deux octets forts en prenant en compte une éventuelle retenue:

| | |
|---|---|
| ld a,(ADR1) | charge l'octet faible d'ADD1 dans l'accumulateur |
| ld hl,ADR2 | charge l'adresse de l'octet faible d'ADD2 dans le registre hl |
| add a,(hl) | additionne les octets faibles entre eux |
| ld (RES),a | stocke l'octet faible du résultat en RES |
| ld a,(ADR1-1) | charge l'octet fort d'ADD1 dans l'accumulateur |
| dec hl | ôte 1 de hl |
| adc a,(hl) | additionne les octets forts et la retenue |
| ld (RES-1),a | stocke l'octet fort du résultat en RES-1 |

L'instruction dec hl a le même effet que ld hl,ADR-1. En effet cette instruction diminue de 1 le contenu de hl qui de ADR1 devient donc bien ADR1-1. L'avantage de l'instruction dec est qu'elle ne prend qu'un octet et est donc plus élégante et plus rapide.

L'instruction adc a,(hl) a le même effet que add a,(hl) si ce n'est qu'elle prend en compte en outre la retenue qui a pu se produire lors de la dernière opération.

Voici maintenant à nouveau notre programme avec les codes hexadécimaux que vous devez entrer:

```
ld a,(&3001)          &3A &1 &30
ld hl,&3003           &21 &3 &30
add a,(hl)            &86
ld (&3005),a          &32 &5 &30
ld a,(&3000)          &3A 00 &30
dec hl                &2B
adc a,(hl)            &8E
ld (&3004),a          &32 &4 &30
RET                   &C9
```

Entrez ce programme avec le petit programme BASIC que nous vous avons fourni après avoir ainsi modifié la ligne 10:

```
10 FOR N=&2000 TO &2012
```

Entrez maintenant en mémoire les valeurs à additionner en suivant le format octet fort-octet faible. Par exemple, pour entrer &5F4C et &8B5A:

```
POKE &3000,&5F:POKE &3001,&4C:POKE &3002,&8B:POKE &3003,&5A
```

Entrez ensuite CALL (&2000) puis:

```
PRINT PEEK(&3004)*256+PEEK(&3005)
```

Vous voyez maintenant apparaître le résultat, 60070 dans notre exemple.

Lors d'une addition sur 8 bits dont le résultat est supérieur à &FF, nous avons donc vu qu'une retenue se produit qui est placée

dans le flag CARRY. Ce flag est un bit du registre f, un registre que nous n'avons pas encore évoqué. Ce registre contient toutes les informations essentielles dont le microprocesseur a besoin. Le flag carry est le bit 0 du registre f.

Passons à une autre application en langage-machine: nous allons essayer de placer des points sur l'écran. Nous pouvons le faire en BASIC soit avec l'instruction PLOT soit en pokant les points directement dans la mémoire écran:

```
10 MODE 2
20 FOR N=&C000 TO &FFFF
30 POKE N,&FF
40 NEXT
```

Vous voyez que l'écran se remplit, non pas ligne par ligne cependant, mais une ligne sur huit, conformément à la structure des caractères que nous avons décrite plus haut.

Pour bien comprendre la logique du programme en langage-machine que nous allons vous présenter, il faut voir que la boucle ci-dessus peut fort bien être écrite ainsi:

```
10 N=N+1
20 IF N=&FFFF THEN END
```

On pourrait cependant écrire cette boucle en langage-machine même sous sa première forme mais ceci nous entraînerait à expliquer plus de nouvelles instructions que nous ne le souhaitons pour l'instant.

Nous voulons donc charger &FF dans toute une série d'adresses de la mémoire. Nous utiliserons l'instruction ld mais pour ne pas réécrire l'instruction de chargement pour chaque adresse concernée, il nous faut créer une boucle, par exemple avec le registre b qui est facile à contrôler.

longueur représentera la longueur de la zone à remplir.
adr1 sera l'adresse de départ du remplissage.

Nous créons la boucle avec: ld b, longueur. Nous plaçons ensuite &FF dans l'accumulateur. En plaçant ADR1 en hl nous créons ce qu'on appelle un "pointeur", qui sera dirigé sur l'adresse de départ du remplissage. Nous chargeons ensuite le contenu de l'accumulateur &FF dans l'adresse indiquée par le pointeur hl. Il nous faut maintenant augmenter hl de 1 (inc hl) pour qu'il pointe sur l'adresse suivante à remplir. Nous devons également diminuer de 1 le registre b (dec b) qui va nous servir de compte à rebours du nombre d'adresses à remplir.
L'instruction suivante devra être un saut conditionnel correspondant au BASIC IF... THEN GOTO... Il nous faut en effet sauter à l'instruction ld (hl),a chaque fois que hl a été incrémenté. Le saut doit se faire tant que le registre b sera différent de 0, et donc tant qu'il restera des adresses à remplir. L'instruction adéquate est jp nz, adresse. jp signifie saut; nz signifie non zero; le saut se produira donc tant que le résultat de la décrémentation du registre b sera non nul. Notre programme doit être bien sûr terminé par un RET:

```
ld b,&C0              &6 &C0
ld a,0                &3E &FF
```

```
ld hl,&C000            &21 &0 &C0
ld (hl),a              &77
inc hl                 &23
dec b                  &5
jp nz,&2007            &C2 &7 &20
RET                    &C9
```

Entrez les codes avec le même programme que tout à l'heure, avec une boucle allant de &2000 à &200D.

Avant de lancer le programme avec CALL &2000, entrez MODE 2, même si vous vous trouvez déjà dans ce mode.

Vous voyez que trois lignes ont été dessinées par notre routine.

L'instruction de saut jp (jump) saute à l'adresse où figure l'instruction ld (hl),a. Ceci est rendu techniquement possible par un autre registre très important du microprocesseur, le PC (programmcounter) ou compteur de programme. Ce registre contient l'adresse de la prochaine instruction à exécuter. L'instruction jp écrit donc dans ce registre l'adresse à laquelle on veut sauter.
Une autre instruction de saut, l'instruction jr procède de manière légèrement différente: cette instruction soustrait en effet ou additionne au contraire suivant le cas le paramètre qui lui est affecté au compteur de programme. C'est ce qu'on appelle un saut relatif: jr e entraînera un saut à l'adresse actuelle plus ou moins le paramètre e. Ce chapitre n'étant qu'une introduction, nous nous cantonnerons ici à ces explications sommaires du fonctionnement des instructions de saut.

JEUX DE CARACTERES ET D'INSTRUCTIONS

Voici une table des caractères et des mnémoniques de votre CPC avec les codes hexadécimaux et décimaux correspondants.

| Code | Caract. | Hex | Assembleur Z 80 | d'après CBh | d'après EDh |
|---|---|---|---|---|---|
| 0 | Null | 00 | nop | rlc b | |
| 1 | SOH | 01 | ld bc,NN | rlc c | |
| 2 | STX | 02 | ld (bc),a | rlc d | |
| 3 | ETX | 03 | inc bc | rlc e | |
| 4 | EOT | 04 | inc b | rlc h | |
| 5 | ENQ | 05 | dec b | rlc l | |
| 6 | ACK | 06 | ld b,N | rlc (hl) | |
| 7 | BEL | 07 | rlca | rlc a | |
| 8 | BS | 08 | ex af,af" | rrc b | |
| 9 | HT | 09 | add hl,bc | rrc c | |
| 10 | LF | OA | ld a,(bc) | rrc d | |
| 11 | VT | OB | dec bc | rrc e | |
| 12 | FF | OC | inc c | rrc h | |
| 13 | CR | OD | dec c | rrc l | |
| 14 | SO | OE | ld c,N | rrc (hl) | |
| 15 | SI | OF | rrca | rrc a | |
| 16 | DLE | 10 | djnz DIS | rl b | |
| 17 | DC 1 | 11 | ld de,NN | rl c | |
| 18 | DC 2 | 12 | ld (de),a | rl d | |
| 19 | DC 3 | 13 | inc de | rl e | |
| 20 | DC 4 | 14 | inc d | rl h | |
| 21 | NAK | 15 | dec d | rl l | |
| 22 | SYN | 16 | ld d,N | rl (hl) | |
| 23 | ETB | 17 | rla | rl a | |
| 24 | CAN | 18 | jr DIS | rr b | |
| 25 | EM | 19 | add hl,de | rr c | |
| 26 | SUB | 1A | ld a,(de) | rr d | |
| 27 | ESC | 1B | dec de | rr e | |
| 28 | FS | 1C | inc e | rr h | |

| | | | | | |
|---|---|---|---|---|---|
| 29 | GS | 1D | dec e | rr l | |
| 30 | RS | 1E | ld e,N | rr (hl) | |
| 31 | US | 1F | rra | rr a | |
| 32 | SP | 20 | jr nz,DIS | sla b | |
| 33 | ! | 21 | ld hl,NN | sla c | |
| 34 | " | 22 | ld(NN),hl | sla d | |
| 35 | CHR$(35) | 23 | inc hl | sla e | |
| 36 | $ | 24 | inc h | sla h | |
| 37 | % | 25 | dec h | sla l | |
| 38 | & | 26 | ld,h,N | sla (hl) | |
| 39 | Chr$(39) | 27 | daa | sla a | |
| 40 | ( | 28 | jr z,DIS | sra b | |
| 41 | ) | 29 | add hl,hl | sra c | |
| 42 | * | 2A | ld hl,(NN) | sra d | |
| 43 | + | 2B | dec hl | sra e | |
| 44 | Chr$(45) | 2C | inc l | sra h | |
| 45 | - | 2D | dec l | sra l | |
| 46 | . | 2E | ld l,N | sra (hl) | |
| 47 | / | 2F | cpl | sra a | |
| 48 | 0 | 30 | jr nc,DIS | | |
| 49 | 1 | 31 | ld sp,NN | | |
| 50 | 2 | 32 | ld (NN),a | | |
| 51 | 3 | 33 | inc sp | | |
| 52 | 4 | 34 | inc (hl) | | |
| 53 | 5 | 35 | dec (hl) | | |
| 54 | 6 | 36 | ld (hl),N | | |
| 55 | 7 | 37 | scf | | |
| 56 | 8 | 38 | jr c,DIS | srl b | |
| 57 | 9 | 39 | add hl,sp | srl c | |
| 58 | : | 3A | ld a,(NN) | srl d | |
| 59 | ; | 3B | dec sp | srl e | |
| 60 | Chr$(60) | 3C | inc a | srl h | |
| 61 | = | 3D | dec a | srl l | |
| 62 | Chr$(62) | 3E | ld a,N | srl (hl) | |
| 63 | ? | 3F | ccf | srl a | |
| 64 | Chr$(64) | 40 | ld b,b | bit 0,b | in b,(c) |
| 65 | A | 41 | ld b,c | bit 0,c | out (c),b |

| | | | | | |
|---|---|---|---|---|---|
| 66 | B | 42 | ld b,d | bit 0,d | sbc hl,bc |
| 67 | C | 43 | ld b,e | bit 0,e | ld(NN),bc |
| 68 | D | 44 | ld b,h | bit 0,h | neg |
| 69 | E | 45 | ld b,l | bit 0,l | retn |
| 70 | F | 46 | ld b,(hl) | bit 0,(hl) | im 0 |
| 71 | G | 47 | ld b,a | bit 0,a | |
| 72 | H | 48 | ld c,b | bit 1,b | inc c,(c) |
| 73 | I | 49 | ld c,c | bit 1,c | out (c),c |
| 74 | J | 4A | ld c,d | bit 1,d | adc hl,bc |
| 75 | K | 4B | ld c,e | bit 1,e | ld bc,(NN) |
| 76 | L | 4C | ld c,h | bit 1,h | |
| 77 | M | 4D | ld c.l | bit 1,l | reti |
| 78 | N | 4E | ld c,(hl) | bit 1,(hl) | |
| 79 | O | 4F | ld c,a | bit 1,a | |
| 80 | P | 50 | ld d,b | bit 2,b | in d,(d) |
| 81 | Q | 51 | ld d,c | bit 2,c | out (c),d |
| 82 | R | 52 | ld d,d | bit 2,d | sbc hl,de |
| 83 | S | 53 | ld d,e | bit 2,e | ld (NN),de |
| 84 | T | 54 | ld d,h | bit 2,h | |
| 85 | U | 55 | ld d,l | bit 2,l | |
| 86 | V | 56 | ld d,(hl) | bit 2,(hl) | im 1 |
| 87 | W | 57 | ld d,a | bit 2,a | ld a,i |
| 88 | X | 58 | ld e,b | bit 2,b | in e,(c) |
| 89 | Y | 59 | ld e,c | bit 3,c | out (c),e |
| 90 | Z | 5A | ld e,d | bit 3,d | adc hl,de |
| 91 | Chr$(91) | 5B | ld e,e | bit 3,e | ld de,(NN) |
| 92 | Chr$(92) | 5C | ld e,h | bit 3,h | |
| 93 | Chr$(93) | 5D | ld e,l | bit 3,l | |
| 94 | | 5E | ld e,(hl) | bit 3,(hl) | im 2 |
| 95 | Chr$(95) | 5F | ld e,a | bit 3,a | |
| 96 | Chr$(96) | 60 | ld h,b | bit 4,b | in h,(c) |
| 97 | a | 61 | ld h,c | bit 4,c | out (c),h |
| 98 | b | 62 | ld h,d | bit 4,d | sbc hl,hl |
| 99 | c | 63 | ld h,e | bit 4,e | ld (NN),hl |
| 100 | d | 64 | ld h,h | bit 4,h | |
| 101 | e | 65 | ld h,l | bit 4,l | |
| 102 | f | 66 | ld h,(hl) | bit 4,(hl) | |

| | | | | | |
|---|---|---|---|---|---|
| 103 | g | 67 | ld h,a | bit 4,a | rrd |
| 104 | h | 68 | ld l,b | bit 5,b | in l,(c) |
| 105 | i | 69 | ld l,c | bit 5,c | out (c),l |
| 106 | j | 6A | ld l,d | bit 5,d | adc hl,hl |
| 107 | k | 6B | ld l,e | bit 5,e | ld de,(NN) |
| 108 | l | 6C | ld l,h | bit 5,h | |
| 109 | m | 6D | ld l,l | bit 5,l | |
| 110 | n | 6E | ld l,(hl) | bit 5,(hl) | |
| 111 | o | 6F | ld l,a | bit 5,a | rld |
| 112 | p | 70 | ld (hl),b | bit 6,b | |
| 113 | q | 71 | ld (hl),c | bit 6,c | sbc hl,sp |
| 114 | r | 72 | ld (hl),d | bit 6,d | ld (NN),sp |
| 115 | s | 73 | ld (hl),e | bit 6,e | |
| 116 | t | 74 | ld (hl),h | bit 6,h | |
| 117 | u | 75 | ld (hl),l | bit 6,l | |
| 118 | v | 76 | halt | bit 6,(hl) | |
| 119 | w | 77 | ld (hl),a | bit 6,a | |
| 120 | x | 78 | ld a,b | bit 7,b | in a,(c) |
| 121 | y | 79 | ld a,c | bit 7,c | out (c),a |
| 122 | z | 7A | ld a,d | bit 7,d | adc hl,sp |
| 123 | | 7B | ld a,e | bit 7,e | ld sp,(NN) |
| 124 | G | 7C | ld a,h | bit 7,h | |
| 125 | R | 7D | ld a,l | bit 7,l | |
| 126 | A | 7E | ld a,(hl) | bit 7,(hl) | |
| 127 | P | 7F | ld a,a | bit 7,a | |
| 128 | H | 80 | add a,b | res 0,b | |
| 129 | I | 81 | add a,c | res 0,c | |
| 130 | C | 82 | add a,d | res 0,d | |
| 131 | | 83 | add a,e | res 0,e | |
| 132 | - | 84 | add a,h | res 0,h | |
| 133 | | 85 | add a,l | res 0,l | |
| 134 | Z | 86 | add a,(hl) | res 0,(hl) | |
| 135 | E | 87 | add a,a | res 0,a | |
| 136 | I | 88 | adc a,b | res 1,b | |
| 137 | C | 89 | adc a,c | res 1,c | |
| 138 | H | 8A | adc a,d | res 1,d | |
| 139 | E | 8B | adc a,e | res 1,e | |

| | | | | | |
|---|---|---|---|---|---|
| 140 | N | 8C | adc a,h | res 1,h | |
| 141 | | 8D | adc a,l | res 1,l | |
| 142 | + | 8E | adc a,(hl) | res 1,(hl) | |
| 143 | | 8F | adc a,a | res 1,a | |
| 144 | T | 90 | sub b | res 2,b | |
| 145 | O | 91 | sub c | res 2,c | |
| 146 | K | 92 | sub d | res 2,d | |
| 147 | E | 93 | sub e | res 2,e | |
| 148 | N | 94 | sub h | res 2,h | |
| 149 | S | 95 | sub l | res 2,l | |
| 150 | | 96 | sub (hl) | res 2,(hl) | |
| 151 | | 97 | sub a | res 2,a | |
| 152 | | 98 | sbc a,b | res 3;b | |
| 153 | | 99 | sbc a,c | res 3,c | |
| 154 | | 9A | sbc a,d | res 3,d | |
| 155 | | 9B | sbc a,e | res 3,e | |
| 156 | | 9C | sbc a,h | res 3,h | |
| 157 | | 9D | sbc a,l | res 3,l | |
| 158 | | 9E | sbc a,(hl) | res 3,(hl) | |
| 159 | | 9F | sbc a,a | res 3,a | |
| 160 | | A0 | and b | res 4,b | |
| 161 | | A1 | and c | res 4,c | cpi |
| 162 | | A2 | and d | res 4,d | ini |
| 163 | | A3 | and e | res 4,e | outi |
| 164 | | A4 | and h | res 4,h | |
| 165 | | A5 | and l | res 4,l | |
| 166 | | A6 | and (hl) | res 4,(hl) | |
| 167 | | A7 | and a | res 4,a | |
| 168 | | A8 | xor b | res 5,b | ldd |
| 169 | | A9 | xor c | res 5,c | cpd |
| 170 | | AA | xor d | res 5,d | ind |
| 171 | | AB | xor e | res 5,e | outd |
| 172 | | AC | xor h | res 5,h | |
| 173 | | AD | xor l | res 5,l | |
| 174 | | AE | xor (hl) | res 5,(hl) | |
| 175 | | AF | xor a | res 5,a | |
| 176 | | B0 | or b | res 6,b | ldir |

| | | | | |
|---|---|---|---|---|
| 177 | B1 | or c | res 6,c | cpir |
| 178 | B2 | or d | res 6,d | inir |
| 179 | B3 | or e | res 6,e | otir |
| 180 | B4 | or h | res 6,h | |
| 181 | B5 | or l | res 6,l | |
| 182 | B6 | or (hl) | res 6,(hl) | |
| 183 | B7 | or a | res 6,a | |
| 184 | B8 | cp b | res 7,b | lddr |
| 185 | B9 | cp c | res 7,c | cpdr |
| 186 | BA | cp d | res 7,d | indr |
| 187 | BB | cp e | res 7,e | otdr |
| 188 | BC | cp h | res 7,h | |
| 189 | BD | cp l | res 7,l | |
| 190 | BE | cp (hl) | res 7,(hl) | |
| 191 | BF | cp a | res 7,a | |
| 192 | C0 | ret nz | set 0,b | |
| 193 | C1 | pop bc | set 0,c | |
| 194 | C2 | jp nz,NN | set 0,d | |
| 195 | C3 | jp NN | set 0,e | |
| 196 | C4 | call nz,NN | set 0,h | |
| 197 | C5 | push bc | set 0,l | |
| 198 | C6 | add a,N | set 0,(hl) | |
| 199 | C7 | rst 0 | set 0,a | |
| 200 | C8 | ret z | set 1,b | |
| 201 | C9 | ret | set 1,c | |
| 202 | CA | jp z,NN | set 1,d | |
| 203 | CB | | set 1,e | |
| 204 | CC | call z,NN | set 1,h | |
| 205 | CD | call NN | set 1,l | |
| 206 | CE | adc a,N | set 1,(hl) | |
| 207 | CF | rst 8 | set 1,a | |
| 208 | D0 | ret nc | set 2,b | |
| 209 | D1 | pop de | set 2,c | |
| 210 | D2 | jp nc,NN | set 2,d | |
| 211 | D3 | out N,a | set 2,e | |
| 212 | D4 | call nc,NN | set 2,h | |
| 213 | D5 | push de | set 2,l | |

| | | | |
|---|---|---|---|
| 214 | D6 | sub N | set 2,(hl) |
| 215 | D7 | rst 16 | set 2,a |
| 216 | D8 | ret c | set 3,b |
| 217 | D9 | exx | set 3,c |
| 218 | DA | jp c,NN | set 3,d |
| 219 | DB | in a,N | set 3,e |
| 220 | DC | call c,NN | set 3,h |
| 221 | DD | | set 3,l |
| 222 | DE | sbc a,N | set 3,(hl) |
| 223 | DF | rst 24 | set 3,a |
| 224 | E0 | ret po | set 4,b |
| 225 | E1 | pop hl | set 4,c |
| 226 | E2 | jp po,NN | set 4,d |
| 227 | E3 | ex (sp),hl | set 4,e |
| 228 | E4 | call po,NN | set 4,h |
| 229 | E5 | push hl | set 4,l |
| 230 | E6 | and N | set 4,(hl) |
| 231 | E7 | rst 32 | set 4,a |
| 232 | E8 | ret pe | set 5,b |
| 233 | E9 | jp (hl) | set 5,c |
| 234 | EA | jp pe,NN | set 5,d |
| 235 | EB | ex de,hl | set 5,e |
| 236 | EC | call pe,NN | set 5,h |
| 237 | ED | | set 5,l |
| 238 | EE | xor N | set 5,(hl) |
| 239 | EF | rst 40 | set 5,a |
| 240 | F0 | ret p | set 6,b |
| 241 | F1 | pop af | set 6,c |
| 242 | F2 | jp p,NN | set 6,d |
| 243 | F3 | di | set 6,e |
| 244 | F4 | call p,NN | set 6,h |
| 245 | F5 | push af | set 6,l |
| 246 | F6 | or N | set 6,(hl) |
| 247 | F7 | rst 48 | set 6,a |
| 248 | F8 | ret m | set 7,b |
| 249 | F9 | ld sp,hl | set 7,c |
| 250 | FA | jp m,NN | set 7,d |
| 251 | FB | ei | set 7,e |
| 252 | FC | call m,NN | set 7,h |
| 253 | FD | | set 7,l |
| 254 | FE | cp N | set 7,(hl) |
| 255 | FF | rst 56 | set 7,a |

## 4.4 LA MEMOIRE DU CPC 464

Il est important de connaître la structure de la mémoire de votre ordinateur, de façon à ce que vous puissiez en tirer le maximum.

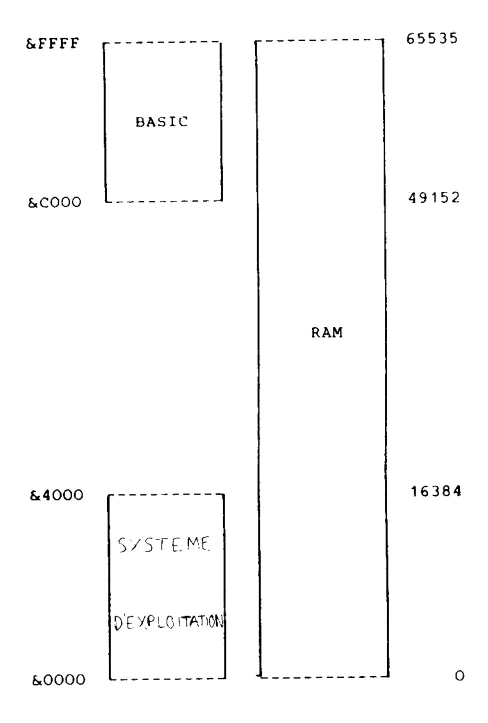
Le CPC 464 est doté d'un microprocesseur Z 80 qui est un processeur 8 bits avec 16 canaux d'adresses qui lui permettent d'adresser 2↑16 = 65536 cases mémoire différentes, chacune comportant 8 bits et pouvant donc contenir 2↑8 = 256 valeurs différentes.

Le CPC 464 contient 64 K octets de RAM et 32 K octets de ROM. Comme le processeur ne peut adresser que 64 K octets un commutateur permet au processeur de choisir s'il veut à un moment donné adresser la ROM ou la RAM qui figurent aux mêmes adresses. Il peut ainsi utiliser pleinement la capacité réelle de 96 K de votre ordinateur. Toutefois, dans les opérations d'écriture, le processeur choisit automatiquement la RAM puisqu'il n'est par définition pas possible d'écrire dans la ROM.

Le CPC 464 possède un composant de la ROM qui contient l'ensemble des 32 K octets dont celle-ci se compose. Les 16 K octets inférieurs constituent le système d'exploitation (de 0 à &3FFF - 16383). Les 16 K octets supérieurs constituent l'interpréteur BASIC qui est placé de &C000 à &FFFF (49152 à 65535).

Voici une figure illustrant ce principe:

CARTE DE LA MEMOIRE DU CPC 464



### 4.4.1 L'OCCUPATION DE LA MEMOIRE PAR LE BASIC ET LE SYSTEME D'EXPLOITATION

La plus grande partie de la mémoire RAM sert au stockage du programme BASIC en mémoire et de ses variables. Cette zone occupe normalement les adresses &170 (368) à &AB7F (43903). Sa taille est donc de 43536 octets.

Lorsque vous lisez la mémoire avec PEEK, vous obtenez toujours le contenu de la RAM. Il n'est donc pas si facile de lire le contenu de la ROM. Le programme de moniteur que nous vous présentons plus loin vous permettra toutefois, entre autre, de le faire.

Voyons maintenant comment est structurée la mémoire RAM: les 64 premiers octets de cette mémoire sont une copie conforme de la même zone de la ROM qui contient les routines restart qui peuvent être appelées par une instruction sur un octet. Ces 8 vecteurs RST servent sur le CPC 464 à appeler des routines de la RAM ou de la ROM. Ils choisissent automatiquement la bonne mémoire. La RAM contient une copie de ces routines de façon à ce que celles-ci puissent être aussi bien exécutées lorsque la ROM est déconnectée que lorsqu'elle est connectée.

Comme ces routines RST sont abondamment utilisées par le système d'exploitation ainsi que par l'interpréteur BASIC, nous allons en décrire quelques-unes. Vous trouverez d'autre part dans notre programme de moniteur une utilisation de la routine RST 3 qui permet de lire le contenu de la ROM.

## 4.5 LES INSTRUCTIONS RST

### RST 0 Adresse &0000

Ce vecteur provoque un reset complet de l'ordinateur de même que la mise sous tension de l'appareil ou le fait d'appuyer simultanément sur les touches CTRL SHIFT ESC.

```
0000  01 89 7F  LD BC,&7F89
0003  ED 49     OUT (C),C
0005  C3 80 05  JP &0580
```

L'instruction OUT sélectionne la ROM et saute à la routine &0580 de la ROM.

Vous pouvez provoquer un reset en BASIC avec CALL 0, en langage-machine avec CALL 0, JP 0 ou RST 0.

### RST 1 Adresse &0008

Cette instruction permet d'appeler une routine du système d'exploition ou de la RAM qui le double. Comme 14 bits suffisent pour la zone entre 0 et &3FFF, les deux autres bits servent à sélectionner la RAM ou la ROM:

Bit 14 = 0 => sélection du système d'exploitation
Bit 14 = 1 => sélection de la RAM

Bit 15 = 0 => sélection de la ROM BASIC

Bit 15 = 1 => sélection de la RAM

Par exemple, pour appeler la routine du système d'exploitation &0826, nous aurions:

```
RST 1
DW  &0826 + &8000
```

Le bit 15 est mis par + &8000, ce qui sélectionne la zone entre &C000 et &FFFF, alors que le bit 14 est à 0, ce qui sélectionne le système d'exploitation au lieu de la RAM. Le code de l'adresse 8 est uniquement un saut en &B982.

RST 2 Adresse &0010

Cette routine restart se compose d'un saut en &BA16 et permet d'appeler une routine dans une ROM d'extension. Après l'instruction RST 2 doit figurer l'adresse de la routine -&C000, c'est-à-dire l'adresse relative de cette routine par rapport au début de la ROM. Les deux bits supérieurs du paramètre permettent de sélectionner quatre ROMs différentes.

RST 3 Adresse &0018

Cette routine comporte un saut en &B98F. Elle permet d'appeler une routine figurant en n'importe quel endroit de la RAM ou de la ROM. Cette instruction doit être utilisée avec un bloc de paramètres

comportant trois octets. Les deux premiers octets indiquent l'adresse de la routine et le troisième l'état ROM/RAM souhaité. Les valeurs 0 à 251 permettent d'appeler la ROM supplémentaire correspondante alors que les 4 valeurs restantes ont les fonctions suivantes:

| Valeur | &0000-&3FFF | &C000-&FFFF |
|---|---|---|
| 252 | Système d'exploitation | BASIC |
| 253 | Système d'exploitation | RAM |
| 254 | RAM | BASIC |
| 255 | RAM | RAM |

RST 4 Adresse &0020

Cette routine comporte un saut en &BACB. Elle permet de lire à partir d'un programme en langage-machine le contenu de la RAM, quel que soit l'état choisi de la ROM. L'instruction RST 4 remplace l'instruction LD A,(HL)

HL doit donc contenir l'adresse de la case de la mémoire RAM à lire.

RST 5 Adresse &0028

Cette routine comporte un saut en &BA2E. Elle permet de sauter à une routine du système d'exploitation. L'adresse doit être placée à la suite de cette instruction.

RST 6 Adresse &0030

Cette instruction RST n'est pas utilisée par le système d'exploitation et peut donc être librement définie par l'utilisateur, par exemple, dans un debugger, pour placer un point d'arrêt (breakpoint).

RST 7 Adresse &0038

Cette routine comporte un saut en &B939. Comme le Z 80 fonctionne en mode d'interruption 1, l'apparition d'une interruption provoque un RST 7. Cette instruction traite l'interruption.

## 4.6 MINI-MONITEUR

Voici maintenant un programme de moniteur qui vous permet de consulter les contenus de la mémoire, de les modifier ou de les stocker sur cassette. Une particularité intéressante de notre programme est qu'il vous permet d'accéder à la ROM BASIC et à la ROM système d'exploitation. Ceci se produit à travers les instructions RST.

Voici maintenant le listing suivi de la description du programme:

```
1 PRINT "DATA BECKER"
100 MEMORY &9FFF:INK 0,1:INK 1,24:INK 2,23:INK 3,15,24:MODE 1
110 n=6:FOR i=0 TO n:READ cmd$(i):NEXT:DATA a,m,s,t,g,r,x
120 GOSUB 830
130 PEN 2:PRINT TAB(10)"M i n i - m o n i t e u r"CHR$(13)TAB(10);:PEN 1:PRINTC
HR$(22)CHR$(1)"___________________________":PRINT:PRIN
T
140 PRINT:PRINT"I n s t r u c t i o n s:"CHR$(13)CHR$(22)CHR$(1)"_______________
_________"CHR$(22)CHR$(0):PRINT
150 PRINT" m = editer zone de la memoire
160 PRINT" a = modifier contenu de la memoire
170 PRINT" s = sauvegarder une zone de la memoire
180 PRINT" t = entrer un texte ASCII
190 PRINT" g = executer un Programme en langage-        machine
200 PRINT" r = selectionner ROM/RAM
210 PRINT" x = retour au BASIC
220 PRINT
230 PRINT:PRINT"Sortie sur ";
240 PEN 3:PRINT"E";:PEN 1:PRINT"cran":PRINT TAB(9)"ou ";:PEN 3:PRINT"I";:PEN 1:P
RINT"mPrimante ?";
250 WHILE a$<>"e" AND a$<>"i":a$=INKEY$:WEND
260 INK 3,24:rom=0
270 PRINT:IF a$="d" THEN 1=16:c=8:GOTO 310
280 PRINT:PRINT:PRINT"Mode ecran ?
290 WHILE a$<>"1" AND a$<>"2":a$=INKEY$:WEND
300 a=VAL(a$):L=a*8:c=0:IF a<>1 THEN MODE a
310 PRINT:PRINT"Instruction ?
320 a$=INKEY$:FOR i=0 TO n:IF a$=cmd$(i) THEN PRINT:ON i+1 GOSUB 490,540,590,700
,750,780,940:GOTO 310
330 NEXT:GOTO 320
```

```
340 '
350 IF a>b THEN RETURN
360 PRINT£c,HEX$(a,4)"  ";
370 FOR i=1 TO l: IF rom THEN GOSUB 890:ELSE x=PEEK(a)
380 PRINT£c,HEX$(x,2)" ";:a=a+1
390 NEXT:PRINT£c," ";:a=a-l
400 IF c=0 THEN 450
410 FOR i=1 TO l: IF rom THEN GOSUB 890:ELSE x=PEEK(a)
420 x=x AND &7F
430 IF x<32 OR x=127 THEN x=46
440 PRINT£c,CHR$(x);:a=a+1:NEXT:PRINT£C:GOTO 350
450 IF L=8 THEN PEN 2
460 FOR I=1 TO L:IF ROM THEN GOSUB 890:ELSE X=PEEK(A)
470 PRINT CHR$(1)CHR$(X);:A=A+1:NEXT:PRINT:PEN 1:GOTO 350
480 '
490 INPUT"adresse ";x$:GOSUB 560:a=x
500 PRINT HEX$(a,4)":";
510 INPUT x$:a$=LEFT$(x$,1):IF a$<"0" OR a$>"9" AND a$<"a" OR a$>"f" THEN RETURN
520 GOSUB 560:POKE a,x:a=a+1:GOTO 500
530 '
540 GOSUB 650:GOSUB 350:RETURN
550 '
560 x=VAL("&"+x$):IF x<0 THEN x=x+2^16
570 RETURN
580 '
590 PRINT"Sauvegarder: ";
600 GOSUB 650
610 INPUT"Nom ";a$
620 SAVE a$,b,a,b-a
630 RETURN
640 '
650 INPUT"de - a ";a$,b$
660 x$=a$:GOSUB 560:a=x
670 x$=b$:GOSUB 560:b=x
680 RETURN
```

```
690
700 INPUT"Adresse ";x$:GOSUB 560
710 LINE INPUT"Entrer texte: ";a$
720 IF LEN(a$)>0 THEN FOR i=1 TO LEN(a$):POKE x-1+i,ASC(MID$(a$,i,1)):NEXT
730 RETURN
740 '
750 INPUT"Adresse ";x$
760 GOSUB 560:CALL x:RETURN
770 '
780 PRINT:INK 3,15,24
790 PRINT"Selectionnez: ";:PEN 3:IF rom THEN PRINT"ROM";:ELSE PRINT"RAM";
800 PEN 1:a$=INKEY$:IF a$=" " THEN rom=1-rom:PRINT CHR$(13);:GOTO 790
810 IF a$=CHR$(13) THEN PRINT:INK 3,24:RETURN:ELSE 800
820 '
830 'charger programme en langage-machine
840 DATA&df,&04,&ab,&c9
850 DATA&07,&ab,&fc
860 DATA&3a,&00,&00,&32,&0e,&ab,&c9
870 FOR i=0 TO &D:READ a:POKE &AB00+i,a:NEXT:RETURN
880 '
890 'lire la ROM
900 ah=INT(a/256):al=a-ah*256
910 POKE &AB08,al:POKE &AB09,ah
920 CALL &AB00
930 x=PEEK(&AB0E):RETURN
940 '
950 CLS:END
```

Description du programme de mini-moniteur:

100 Diminution de la limite supérieure de la RAM BASIC, définition des couleurs et choix du mode écran

110 Lecture des instructions qui sont placées dans cmd$

130-140 Edition du titre. Utilisation du mode transparent pour le soulignage

150-220 Affichage du jeu d'instructions et des lettres permettant de les appeler

230-260 Définition du périphérique de sortie pour l'instruction M

270-300 Suivant ce que vous avez choisi, le nombre d'octets par ligne est fixé à 16 pour une sortie sur imprimante et le numéro de canal à 8. Si vous avez opté pour la sortie sur écran, vous pouvez encore choisir entres les modes 80 et 40 caractères.

310-330 Boucle principale d'attente des entrées. Les caractères entrés sont comparés aux commandes des différentes instructions et, en cas d'identité, on saute à la routine voulue.

350-390 HEX-DUMP: une fois l'adresse entrée, 1 (8 ou 16) octets sont édités sous forme de nombres hexadécimaux. La lecture de la RAM peut se faire simplement par des PEEK mais la lecture de la ROM ne peut se faire que par la routine 890.

400-470 ASCII-DUMP: pour l'édition des caractères ASCII correspondant au contenu d'une zone de la mémoire, on examine tout d'abord si la sortie doit se faire sur l'écran ou sur l'imprimante. Pour une sortie sur imprimante, on supprime le bit supérieur du caractère. Lorsqu'il s'agit d'un caractère de contrôle, celui-ci est remplacé par un point. Pour la sortie sur écran, tous les caractères sont représentés grâce à CHR$(1).

490-520 Cette routine permet de modifier le contenu de la mémoire. Après que vous ayez entré une adresse, vous pouvez entrer son nouveau contenu. L'adresse suivante est alors automatiquement affichée et on attend votre nouvelle entrée. Vous pouvez sortir de ce mode en entrant un point.

540 Les deux appels de sous-programmes traitent l'instruction M. 650 vous permet d'entrer les adresses de début et de fin de la zone à afficher et la seconde

sous-routine réalise l'affichage.

560-570 Cette routine transforme une suite de chiffres hexadécimaux en une valeur numérique et vous épargne ainsi d'avoir à utiliser le '&'.

590-630 Cette routine vous permet d'effectuer une sauvegarde d'une zone de la mémoire sur cassette. Vous devez entrer les limites de la zone à sauvegarder ainsi que le nom que vous souhaitez lui donner.

650-680 Cette routine vous permet d'entrer les limites d'une zone de la mémoire qui sont placées dans les variables a et b.

700-730 Cette routine vous permet d'entrer un texte directement en mémoire. La ligne 720 poke le texte caractère par caractère en mémoire.

750-760 Ces deux lignes vous permettent de faire exécuter un programme en langage-machine se trouvant en mémoire. Vous indiquez l'adresse du programme à exécuter.

780-810 Cette routine vous permet de sélectionner entre ROM et RAM. L'état actuellement sélectionné clignote. Pour passer de la ROM à la RAM et vice versa, il vous suffit d'appuyer sur la touche espace. Une fois l'état voulu sélectionné, appuyez sur ENTER.

830-870 Le programme en langage-machine permettant de lire la

ROM figure en lignes de DATA. La ligne 870 place ce programme dans l'endroit voulu de la mémoire.

890-930 Ici se produit la lecture d'un octet de la ROM. L'adresse de l'octet à lire est placée dans la variable a. Cette adresse, divisée en octets fort et faible est placée dans la routine en langage-machine. Cette routine place le résultat de la lecture en &AB0E où il peut être lu par un PEEK.

950 X provoque l'effaçage de l'écran et la fin du programme.

Pour lire la ROM, nous utilisons une routine du système d'exploitation que nous appelons grâce à RST 3. La définition de l'octet à lire se fait simplement en pokant les octets fort et faible de l'adresse correspondante à la suite de l'instruction ld, à l'adresse READROM. Ce n'est qu'ensuite que la routine est appelée.

Voici maintenant deux exemples d'édition par le moniteur, le premier vous montre un exemple de contenu de la RAM (programme BASIC), le second un exemple de contenu de la ROM.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0160 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ........ |
| 0168 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ........ |
| 0170 | 31 | 00 | 64 | 00 | AA | 20 | 1C | FF | 1.d.* .. |
| 0178 | AA | 01 | 20 | A2 | 20 | 0E | 2C | 0F | *. " .,. |
| 0180 | 01 | 20 | A2 | 20 | 0F | 2C | 19 | 18 | . " .,.. |
| 0188 | 01 | 20 | A2 | 20 | 10 | 2C | 19 | 17 | . " .,.. |
| 0190 | 01 | 20 | A2 | 20 | 11 | 2C | 19 | 0F | . " .,.. |
| 0198 | 2C | 19 | 18 | 01 | 20 | AD | 20 | 0F | ,... - . |
| 01A0 | 00 | 41 | 00 | 6E | 00 | 0D | 05 | 00 | .A.n.... |
| 01A8 | EE | EF | 14 | 20 | 01 | 20 | 9E | 20 | no. . . |
| 01B0 | 0D | 0E | 00 | E9 | EF | 0E | 20 | EC | ...io. l |
| 01B8 | 20 | 0D | 05 | 00 | EE | 01 | 20 | C3 | ...n. C |
| 01C0 | 20 | 03 | 09 | 00 | 63 | 6D | E4 | 28 | ...cmd( |
| 01C8 | 0D | 0E | 00 | E9 | 29 | 01 | 20 | B0 | ...i). 0 |
| 01D0 | 01 | 20 | 8C | 20 | 61 | 2C | 6D | 2C | . . a,m, |
| 01D8 | 73 | 2C | 74 | 2C | 67 | 2C | 72 | 2C | s,t,g,r, |
| 01E0 | 78 | 00 | 0A | 00 | 78 | 00 | 9F | 20 | x...x.. |
| 01E8 | 1D | 06 | 0B | 00 | 6B | 00 | 82 | 00 | ....k... |
| 01F0 | BB | 20 | 10 | 01 | 20 | BF | 20 | EA | ; .. ? j |
| 01F8 | 20 | 28 | 19 | 0A | 29 | 20 | 22 | 4D | (..) "M |
| 0200 | 20 | 69 | 20 | 6E | 20 | 69 | 20 | 2D | i n i - |

```
C000  80 01 00 00 4C C0 31 00   ....L@1.
C008  C0 CD CB BC CD C4 F4 DA    MK<MDtZ
C010  00 00 21 00 AC 36 00 06   ..!.,6..
C018  1B 23 36 C9 10 FB 21 3F   .#6I.;!?
C020  C0 CD 37 C3 AF 32 00 AC   @M7C/2.,
C028  CD CB DD CD 84 CA CD 97   MKäM.JM.
C030  BD CD D3 C0 CD 3E C1 11   =MS@M>A.
C038  F0 00 CD 06 F7 18 25 20   p.M.w.%
C040  42 41 53 49 43 20 31 2E   BASIC 1.
C048  30 0A 0A 00 42 41 53 49   0...BASI
C050  C3 00 CD E1 CE C0 31 00   C.MaN@1.
C058  C0 CD 9A E7 CD 63 E1 CD   @M.gMcaM
C060  43 CA 38 54 CD 01 AC 31   CJ8TM.,1
C068  00 C0 CD 62 C1 CD D6 DD   .@MbAMVä
C070  DC B6 BC CD 48 BB CD 86   ö6<MH;M.
C078  C3 3A 45 AE B7 C4 3E C1   C:E.7D>A
```

# CHAPITRE 5 LIGNES BASIC, VARIABLES ET TOKENS

## 5.1 FORMAT D'UNE LIGNE BASIC EN MEMOIRE

Pour suivre ce chapitre, vous allez avoir besoin du programme de moniteur du chapitre 4. Le format des lignes BASIC ainsi que celui des variables vous montrera bien comment votre ordinateur passe du langage-machine à un langage évolué tel que le BASIC.

Votre ordinateur ne peut en effet comprendre directement que des codes binaires. C'est le système d'exploitation ainsi que l'interpréteur BASIC qui transforment toutes les instructions BASIC en code binaire. Si vous écrivez la ligne BASIC:

```
1 PRINT"DATA BECKER"
```

votre ordinateur ne peut rien comprendre directement. Même avec l'interpréteur, votre ordinateur ne comprend pas tout et il vous le fait savoir, chaque fois qu'il affiche un 'SYNTAX ERROR'.

Voyons tout de suite un exemple pratique: ajoutez au programme de mini-moniteur la ligne 1 PRINT "DATA BECKER". Lancez le programme et faites afficher la zone entre 170 et 183. Voici ce que vous obtenez (nous avons transcrit les résultats en décimal):

| ADRESSE | VALEUR | COMMENTAIRE |
|---|---|---|
| 368 | 20 | octet faible de la longueur de ligne |
| 369 | 0 | octet fort de la longueur de ligne |
| 370 | 1 | octet faible du numéro de ligne |
| 371 | 0 | octet fort du numéro de ligne |
| 372 | 191 | token pour PRINT |
| 373 | 32 | valeur ASCII pour espace |
| 374 | 34 | valeur ASCII pour guillemet |
| 375 | 68 | valeur ASCII pour "D" |
| 376 | 65 | valeur ASCII pour "A" |
| 377 | 84 | valeur ASCII pour "T" |
| 378 | 65 | valeur ASCII pour "A" |
| 379 | 32 | valeur ASCII pour espace |
| 380 | 66 | valeur ASCII pour "B" |
| 381 | 69 | valeur ASCII pour "E" |
| 382 | 67 | valeur ASCII pour "C" |
| 383 | 75 | valeur ASCII pour "K" |
| 384 | 69 | valeur ASCII pour "E" |
| 385 | 82 | valeur ASCII pour "R" |
| 386 | 34 | valeur ASCII pour guillemet |
| 387 | 0 | code de séparation |

Examinons maintenant la signification de tous ces codes: la longueur de la ligne BASIC est exprimée par deux octets:

20 (octet faible) + 256 * 0 (octet fort) = 20

Cette longueur de ligne correspond bien aux adresses dont nous avons exposé le contenu ci-dessus.

Les deux octets suivants indiquent le numéro de ligne 1. Ensuite vient un token dont la signification vous sera expliquée dans la section suivante; puis viennent les codes ASCII pour le texte "DATA BECKER".

Toutes les lignes de BASIC suivent ce format avec une exception notable cependant en ce qui concerne les lignes BASIC comportant une définition de variable. Nous reviendrons plus loin sur cette particularité.

## 5.2 LES TOKENS

Le terme TOKEN signifie signe. Il désigne la valeur qui est affectée à chaque instruction BASIC de même qu'un code binaire est affecté à chaque instruction du langage-machine. L'intérêt essentiel du token est un gain de place en mémoire BASIC. Si vous utilisez par exemple l'instruction PRINT dans un programme, l'interpréteur BASIC remplacera les 5 lettres -5 octets- qui la composent, chaque fois qu'il les rencontrera (sauf si elles font partie d'un texte entre guillemets) par un code d'un octet appelé token. C'est d'ailleurs pourquoi il ne faut jamais utiliser les noms des instructions BASIC pour définir des variables. L'interpréteur les remplacera en effet toujours par le token de l'instruction correspondante.

Voici maintenant une liste de tous les tokens n'utilisant qu'un octet. Cette liste n'est donc pas tout à fait complète puisqu'il manque les tokens de deux octets.

Une dernière remarque cependant: la connaissance des tokens vous permet de faire modifier un programme BASIC par lui-même. Vous pouvez ainsi user d'un petit truc pour interdire par exemple que des utilisateurs mal intentionnés de vos programmes ne suppriment la ligne de COPYRIGHT qui vous permet de revendiquer la propriété de votre travail.

Une ligne de copyright se présente ainsi:

```
1 REM "COPYRIGHT 1985 BY DATA BECKER
```

Si vous placez maintenant dans le seconde ligne de votre programme une instruction essentielle sans laquelle le programme ne peut pas fonctionner (par exemple une instruction DIM), vous pouvez ensuite insérer au milieu de votre programme une ligne POKE 372,191 qui remplacera le token du REM de la première ligne par le token de l'instruction PRINT. Si la ligne de copyright n'a pas été supprimée, tout se passera bien. Par contre si cette ligne manque, c'est la seconde ligne avec l'instruction essentielle au bon fonctionnement du programme qui se trouvera détruite.

| | | | | | |
|---|---|---|---|---|---|
| After | 128 | Mid | 172 | Tron | 211 |
| Auto | 129 | Mode | 173 | Wait | 212 |
| Border | 130 | Move | 174 | Wend | 213 |
| Call | 131 | Mover | 175 | While | 214 |
| Cat | 132 | Next | 176 | Width | 215 |
| Chain | 133 | New | 177 | Window | 216 |
| Chain Merge | 133&171 | On | 178 | Write | 217 |
| Clear | 134 | On Break | 179 | Zone | 218 |
| Clg | 135 | On Error goto | 180 | Di | 219 |
| Close in | 136 | On sq | 181 | Ei | 220 |
| Close out | 137 | Open in | 182 | Erl | 227 |
| Cls | 138 | Open out | 183 | Fn | 228 |
| Cont | 139 | Origin | 184 | Spc | 229 |
| Data | 140 | Out | 185 | Step | 230 |
| Def fn | 141 | Paper | 186 | Swap | 231 |
| Def int | 142 | Pen | 187 | Tab | 234 |
| Def read | 143 | Plot | 188 | Then | 235 |
| Def Str$ | 144 | Plotr | 189 | To | 236 |
| Deg | 145 | Poke | 190 | Using | 237 |
| Delete | 146 | Print | 191 | > | 238 |
| Dim | 147 | Chr$(39) | 192 | = | 239 |
| Draw | 148 | Rad | 193 | >= | 240 |
| Drawr | 149 | Randomize | 194 | < | 241 |
| Edit | 150 | Read | 195 | <> | 242 |
| Else | 151 | Release | 196 | <= | 243 |
| End | 152 | Rem | 197 | + | 244 |
| Ent | 153 | Renum | 198 | - | 245 |
| Env | 154 | Restore | 199 | * | 246 |
| Erase | 155 | Resume | 200 | / | 247 |
| Error | 156 | Return | 201 | ↑ | 248 |
| Every | 157 | Run | 202 | \ | 249 |

| | | | | | |
|---|---|---|---|---|---|
| For | 158 | Save | 203 | And | 250 |
| Gosub | 159 | Sound | 204 | Mod | 251 |
| Goto | 160 | Speed | 205 | Or | 252 |
| If | 161 | Stop | 206 | Xor | 253 |
| Ink | 162 | Symbol | 207 | Not | 254 |
| Input | 163 | Tag | 208 | Abs | 255 |
| Key | 164 | Tag off | 209 | | |
| Key Def | 164&141 | Troff | 210 | | |
| Let | 165 | | | | |
| Line Input | 166&163 | | | | |
| List | 167 | | | | |
| Load | 168 | | | | |
| Locate | 169 | | | | |
| Memory | 170 | | | | |
| Merge | 171 | | | | |

Les valeurs 221 à 226 et 232 et 233 ne sont pas utilisées mais produisent un effet intéressant. Si vous créez en effet une ligne 1 REM et si vous notez bien le numéro de la seconde ligne, vous pouvez encore une fois protéger votre programme contre une utlisation que vous n'avez pas autorisée. Il vous suffit pour celà d'entrer POKE 372,226. Alors les personnes non averties qui lanceraient votre programme verraient défiler de manière ininterrompue le message syntax error in 1. Le seul moyen de retrouver le contrôle de l'ordinateur dans un pareil cas consiste à appuyer simultanément sur les touches CTRL, SHIFT et ESC ce qui provoque un RESET total et détruit donc également le programme en mémoire.

Si vous utilisez cette méthode de protection, il faut que vous fassiez attention à ne pas lancer votre programme par un RUN mais par un RUN numéro de la deuxième ligne. Il ne faut pas non plus qu'il y ait dans votre programme des instructions de saut à la première ligne.

Cette protection est d'autant plus efficace qu'elle empêche également le listage de la ligne où elle figure.

Si vous voulez la supprimer, entrez POKE 372,197.

## 5.3 FORMAT DES VARIABLES

Nous avons choisi 5 cas représentatifs de variables pour vous expliquer le format de ces lignes BASIC d'exception que sont les lignes avec variables. Ici encore vous aurez besoin du moniteur pour suivre nos développements.

Chargez le moniteur puis entrez comme premier exemple la ligne 1 C=100.

Lancez le programme puis examinez la zone de la mémoire comprise entre 170 et 17B. Vous obtenez (en décimal):

| ADRESSE | VALEUR | COMMENTAIRE |
|---|---|---|
| 368 | 12 | octet faible de la longueur de ligne |
| 369 | 0 | octet fort de la longueur de ligne |
| 370 | 1 | octet faible du numéro de ligne |
| 371 | 0 | octet fort du numéro de ligne |
| 372 | 13 | indique une variable numérique |
| 373 | 5 | longueur du nom de la variable +4 |
| 374 | 0 | 0 de séparation |
| 375 | 227 | valeur ASCII du nom de la variable +128 |
| 376 | 239 | token de "=" |
| 377 | 25 | grandeur de la variable |
| 378 | 100 | valeur de la variable |
| 379 | 0 | 0 de séparation |

La signification des 4 premières valeurs est la même qu'au chapitre 5.1.

La valeur 13 en 372 n'est pas un token mais un code indiquant que la variable à suivre est une variable numérique. Pour une variable alphanumérique, nous aurions le code 3.

La valeur en 373 indique la longueur du nom de la variable + 4. Le 0 qui suit n'a aucune signification autre que de séparation ainsi que le 0 en 379.

Le nom de la variable apparaît ensuite, codé ainsi: tous les caractères du nom sont représentés par le code ASCII correspondant sauf le dernier caractère qui est représenté par son code ASCII + 128.

La valeur 239 en 376 est le token du signe égale. Ce n'est donc pas le code ASCII de "=" pour que l'ordinateur sache bien que le signe "=" ne fait pas partie du nom de la variable. La valeur suivante (25) indique la grandeur de la variable. Ici elle signifie que la valeur de la variable peut être codée sur un seul octet et qu'il s'agit d'un nombre entier. Vous trouverez plus loin un tableau des différents codes pour la grandeur des variables.

Le 100 en 378 correspond enfin à la valeur de la variable.

Deuxième exemple, entrez 1 C=1000. Lancez à nouveau le programme et faites afficher la zone entre 174 et 17C. Voici les valeurs décimales correspondantes:

| ADRESSE | VALEUR | COMMENTAIRE |
|---|---|---|
| 372 | 13 | indique une variable numérique |
| 373 | 5 | longueur du nom de la variable +4 |
| 374 | 0 | 0 de séparation |
| 375 | 227 | valeur ASCII du nom de la variable +128 |
| 376 | 239 | token de "=" |
| 377 | 26 | grandeur de la variable |
| 378 | 232 | octet faible de la valeur de la variable |
| 379 | 3 | octet fort de la valeur de la variable |
| 380 | 0 | 0 de séparation |

Rien ne change jusqu'en 376 mais la grandeur de la variable est maintenant représentée par le code 26 qui indique que la valeur de cette variable est supérieure à 255 mais inférieure à 65536 et qu'elle est donc représentée par deux octets et que d'autre part il s'agit d'un nombre entier.

Voyons maintenant comment votre CPC code les variables numériques dont la valeur soit n'est pas un nombre entier, soit est supérieure à 65535. Entrez donc: 1 C=100000

Listez la zone entre 174 et 17F. Le résultat en décimal est:

| ADRESSE | VALEUR | COMMENTAIRE |
|---|---|---|
| 372 | 13 | indique une variable numérique |
| 373 | 5 | longueur du nom de la variable +4 |
| 374 | 0 | 0 de séparation |
| 375 | 227 | valeur ASCII du nom de la variable +128 |

| | | |
|---|---|---|
| 376 | 239 | token de "=" |
| 377 | 31 | grandeur de la variable |
| 378 | 0 | VAR 1 |
| 379 | 0 | VAR 2 |
| 380 | 80 | VAR 3 |
| 381 | 67 | VAR 4 |
| 382 | 145 | VAR 5 |
| 383 | 0 | 0 de séparation |

Le code 31 en 377 indique que la variable n'est pas un nombre entier ou qu'elle est supérieure à 65535. les 5 valeurs suivantes indiquent la valeur de la variable, selon la formule suivante qui est extrêmement complexe et que vous n'êtes donc pas obligé de comprendre à tout prix:

Valeur = ( 2 ↑ (VAR 5 - 145)) * (65536 + (VAR 2/128) +
(VAR 3 * 2) + (VAR 4 * 512) + (VAR 1 / 32800))

## 6.1 LE MANCHE A BALAI UTILISE COMME SOURIS

La souris est un de ces nouveaux concepts apparus pour rendre les programmes informatiques plus agréables et plus faciles à utiliser. La souris est un petit boitier avec une roulette et une touche que vous pouvez déplacer sur un document pendant qu'un symbole, le plus souvent une flèche, suit à l'écran le mouvement de votre souris. Vous pouvez avec ce symbole sélectionner différents champs de l'écran auxquels ont été affectées différentes fonctions que vous pouvez ensuite faire exécuter en appuyant sur le bouton de la souris. La souris remplace donc le clavier de l'ordinateur. Après un peu d'exercice vous pouvez même manipuler la souris beaucoup plus rapidement que le clavier, à moins cependant que vous ne tapiez à la machine avec vos 10 doigts.

Il n'existe pas encore de souris pour le CPC 464, mais comme nous l'avons déjà fait dans notre programme de dessin, nous allons nous servir du joystick (manche à balai) en guise de souris.

Pour que vous compreniez bien le fonctionnement de la souris, nous avons écrit quelques points du menu d'un programme de traitement de texte que vous allez pouvoir sélectionner en amenant avec votre joystick (Amstrad ou compatible Atari) le petit point qui nous sert de curseur dans le rond correspondant au point du menu que vous choisissez.

Faites attention en recopiant le programme à bien taper le bon nombre d'espaces en lignes 30,50,70,90,110,130,150 et 170!

```
souris
5 MODE 2
10 CLS
20 LOCATE 1,1
30 PRINT"Creer le texte   O"
40 LOCATE 40,1
50 PRINT"Examiner texte   O"
60 LOCATE 1,8
70 PRINT"Modifier texte   O"
80 LOCATE 40,8
90 PRINT"Supprimer texte  O"
100 LOCATE 1,17
110 PRINT"Sauver texte     O"
120 LOCATE 40,17
130 PRINT"Charger texte    O"
140 LOCATE 1,25
150 PRINT"Imprimer texte   O"
160 LOCATE 40,25
170 PRINT"Fin de Programme O"
180 a=JOY(0)
190 PLOT x*8,400-y*16,0
200 IF a=1 THEN y=y-1
210 IF y<1 THEN y=25
220 IF a=2 THEN y=y+1
230 IF y>25 THEN y=1
240 IF a=4 THEN x=x-1
250 IF x<1 THEN x=80
260 IF a=8 THEN x=x+1
270 IF x>80 THEN x=1
280 PLOT x*8,400-y*16,1
290 IF a=16 THEN GOTO 300 ELSE GOTO 180
300 IF x=18 AND y=1 THEN GOTO 400
310 IF x=58 AND y=1 THEN GOTO 440
320 IF x=18 AND y=8 THEN GOTO 480
330 IF x=58 AND y=8 THEN GOTO 520
340 IF x=18 AND y=17 THEN GOTO 560
350 IF x=58 AND y=17 THEN GOTO 600
360 IF x=18 AND y=25 THEN GOTO 640
370 IF x=58 AND y=25 THEN GOTO 680
380 GOTO 180
390 END
400 CLS
410 PRINT"Creer le texte"
420 INPUT a$:IF a$<>"w" GOTO 420
430 GOTO 10
440 CLS
450 PRINT"Examiner le texte"
460 INPUT a$:IF a$<>"w" GOTO 460
470 GOTO 10
480 CLS
490 PRINT"Modifier le texte"
500 INPUT a$:IF a$<>"w" GOTO 500
510 GOTO 10
```

Commentaire du programme:

------------------------

10-170 Affichage du menu

180 La valeur du port joystick est placée dans une variable

190 Suppression du petit curseur

200-270 Routine de déplacement du petit curseur en fonction du mouvement du joystick. Le déplacement se fait en modifiant les valeurs des deux variables de coordonnées du joystick. Chaque fois, on doit contrôler si la valeur obtenue ne sort pas du cadre de l'écran pour éviter une erreur IMPROPER ARGUMENT.

280 Fixation du petit curseur

290 Si vous appuyez sur le bouton FIRE, le programme saute à la sous-routine correspondante, sinon le port joystick est a nouveau interrogé.

300-370 Le programme saute ici lorsque vous appuyez sur le bouton FIRE. On examine alors où se trouvait le curseur lorsque vous avez appuyé sur le bouton FIRE.

380 Retour a l'interrogation du port si vous avez appuyé sur le bouton FIRE alors que le curseur ne se trouvait pas dans un "O".

400- Vous pouvez insérer ici les routines de votre programme. Nous n'avons écrit que quelques exemples. Faites attention à adapter les numéros de lignes lors de l'appel des routines.

Apres avoir lancé ce programme, vous pouvez sélectionner un des points du menu qui vous sont proposés là condition que vous ayez ecrit toutes les routines sur le modèle que nous vous

fournissons). Il y a maintenant deux améliorations importantes auxquelles vous pouvez vous attaquer: premièrement agrandir le curseur. Il vous faut pour celà travailler avec l'instruction LOCATE au lieu de travailler avec l'instruction PLOT. Vous pourriez ensuite essayer d'agrandir les ronds de sélection, de façon à ce qu'ils soient plus facile à atteindre. Il faut pour celà interroger en 300-370 une zone plus importante que celle que nous interrogeons. Le mieux est d'utiliser pour celà des instructions d'opération logique.

## 6.2 COMMANDE DU MOTEUR DU LECTEUR DE CASSETTE

Les deux routines dont voici les adresses vous permettent de commander le moteur du lecteur de cassette:

&BC6E    Démarrage du moteur

&BC71    Arrêt du moteur

Pour faire démarrer le moteur il suffit donc d'entrer CALL &BC6E. Vous ne pouvez toutefois utiliser ces routines que si la touche PLAY est enfoncée.

## 6.3 PROTECTION ELEMENTAIRE CONTRE LES COPIES

Si vous tenez à protéger vos programmes contre la copie, vous serez certainement très heureux d'apprendre que le CPC 464 possède un dispositif de protection des programmes intégré!

Il suffit en effet pour protéger un programme d'ajouter au nom du programme, lors de la sauvegarde, une virgule suivie d'un P. Un programme que vous aurez ainsi sauvegardé ne peut être chargé qu'avec l'instruction RUN. Il est alors chargé puis lancé automatiquement. Il n'est pas possible d'en lister le contenu ni de le sauvegarder à nouveau. En effet, si vous interrompez le programme, ce qui est possible, celui-ci se détruit automatiquement.

## 6.4 SAUVEGARDE D'UNE ZONE DE LA MEMOIRE

Nous attirons ici a nouveau votre attention sur deux possibilités tres interessantes dont nous avons déjà fait usage. La première consiste a sauvegarder certaines zones de la mémoire comme nous l'avons fait dans notre programme de moniteur.

La seconde également tres importante est la faculté de sauvegarder tout ou partie de l'écran ainsi que nous l'avons utilisée dans notre éditeur graphique. Ceci vous permet notamment de faire apparaitre un dessin a l'écran beaucoup plus vite qu'avec les instructions BASIC de dessin du CPC 464. Pour plus de détails, nous vous invitons a vous reporter aux chapitres où ces problèmes ont deja été évoqués.

## 6.5 REAFFECTATION DES TOUCHES

Deux instructions KEY et KEY DEF permettent de modifier l'affectation des touches de votre CPC 464. Voici donc un programme qui vous permet de réaffecter les touches du clavier. Ce programme s'explique par lui-même.

```
clavier
10 MODE 2
20 LOCATE 22,5:PRINT"K E Y   M A N A G E R"
30 LOCATE 22,6:PRINT"====================="
40 PRINT:PRINT:PRINT:PRINT:PRINT
50 PRINT"Ce programme vous permet d'affecter au bloc de 10 touches de chiffres d
es        fonctions speciales et de recreer d'autre p
art un clavier AZERTY."
60 PRINT:PRINT"Frappez une touche"
70 CALL &BB18
80 CLS
90 PRINT:PRINT"Les chiffres recoivent l'affectation suivante:"
100 PRINT:PRINT:PRINT
110 KEY 137,d$
120 PRINT"Touche 0 :  LIST"
130 PRINT"Touche 1 :  RUN"
140 PRINT"Touche 2 :  LOAD"
150 PRINT"Touche 3 :  MODE"
160 PRINT"Touche 4 :  SAVE"
170 PRINT"Touche 5 :  EDIT"
180 PRINT:PRINT:PRINT
190 PRINT"Vous pouvez en outre affecter a 5 autres touches une sequence de 15 ca
racteres."
200 INPUT"Pour ce faire, entrez un 'd'";d$
210 IF d$="d" THEN GOTO 280
220 CLS
230 PRINT"Ce programme redefinit en outre le clavier d'apres le format francais
AZERTY."
240 PRINT:PRINT:PRINT:PRINT
250 PRINT"Frappez une touche"
260 CALL &BB18
270 GOTO 350
```

```
280 CLS
290 INPUT"Affectation de la touche 6";a$
300 INPUT"Affectation de la touche 7";b$
310 INPUT"Affectation de la touche 8";c$
320 INPUT"Affectation de la touche 9";d$
330 INPUT"Affectation de la touche .";e$
340 GOTO 220
350 KEY 128,"list"+CHR$(13)
360 KEY 129,"run"+CHR$(13)
370 KEY 130,"load"+CHR$(13)
380 KEY 131,"mode "
390 KEY 132,"save"+CHR$(13)
400 KEY 133,"edit "
410 KEY 133,a$
420 KEY 134,b$
430 KEY 135,c$
440 KEY 136,d$
450 KEY 137,e$
460 KEY DEF 67,1,97,65
470 KEY DEF 59,1,122,90
480 KEY DEF 69,1,113,81
490 KEY DEF 29,1,109,77
500 KEY DEF 71,1,119,87
510 KEY DEF 38,1,58,42
```

## 6.6 ADRESSES D'APPEL DE ROUTINES INTERESSANTES

Nous allons vous présenter dans les pages suivantes des adresses de routines du système d'exploitation que vous pouvez parfaitement intégrer dans votre programme, même si vous n'êtes pas un professionnel de la programmation en langage-machine; il suffit que vous ayez lu l'introduction au langage-machine que constitue le chapitre 4. Nous indiquons pour chaque routine sa fonction ainsi que les registres d'entrée-sortie correspondants ainsi, s'il y a lieu, que certaines particularités de chaque routine. Voici auparavant, à nouveau, le programme BASIC de chargement qui vous permet d'appeler toutes les routines:

```
10 REM CHARGEUR BASIC
20 MEMORY &2FFF:d=0
30 INPUT"VALEUR ";w
40 IF w=0 THEN GOTO 80
50 POKE &3000+d,w
60 d=d+1
70 GOTO 30
80 CALL &3000
```

Le programme de chargement en mémoire des routines d'appel de routines système est légèrement modifié par rapport à celui que nous vous donnions au chapitre 4.3. De cette façon vous pouvez charger n'importe quelle routine sans devoir spécifier chaque fois la longueur de la routine à charger. Le programme ne s'interrompt en effet que lorsque vous entrez un 0. Si vous devez placer un 0 en mémoire, modifiez le test en ligne 40.

Si vous voulez intégrer des routines système dans le fonctionnement de vos programmes, placez les routines d'appel sous forme de lignes de DATA que vous pouvez faire lire et poker en mémoire grâce à des variables BASIC.

### 6.6.1 COLORIER UNE ZONE

Adresse : BC44 (hex)

48196 (déc)

Cette routine permet de colorier des zones de l'écran. La grandeur de la zone dépend du mode écran choisi et des points-limites que vous pouvez choisir librement.

Registres de transmission:
- a code couleur (voyez également le chapitre 2.10 sur la mémoire écran)
- h colonne limite gauche de la zone à colorier
- d colonne limite droite de la zone à colorier
- l ligne limite supérieure de la zone à colorier
- e ligne limite inférieure de la zone à colorier

Faites très attention à ne pas fournir comme valeurs limites des valeurs situées en dehors de l'écran car celà pourrait bien "planter" votre CPC 464. L'angle supérieur gauche de l'écran a la valeur 0,0.

Registres de réponse: - - -

Valeurs hexadécimales pour le chargeur BASIC: 3E, code couleur, 21, limite supérieure, limite gauche, 11, limite inférieure, limite droite, CD, 44, BC, C9, 0

Toutes les valeurs dont nous ne donnons que le nom (code couleur, limites, etc...) peuvent être définies arbitrairement.

Format assembleur:

```
ld a,N
ld hl,NN
ld de,NN
call 44,BC
ret
```

## 6.6.2 MODIFIER LA BANQUE ECRAN

Adresse : BC06 (hex)

48134 (déc)

Cette routine permet de placer la mémoire écran dans l'une quelconque des 4 banques disponibles (0-3). Les seules possibilités intéressantes sont cependant la banque 1 ou la banque standard 3 car la sélection d'une autre banque peut conduire à des conflits avec le système d'exploitation.

Registres de transmission: a il faut placer dans l'accumulateur l'octet significatif (octet fort) de l'adresse où commence la nouvelle banque écran.

Faites très attention à ne pas fournir d'autres valeurs comme paramètre que 00, 40, 80 ou C0 car vous risqueriez d'aboutir à des résultats désagréables.

Registres de réponse: - - -

Valeurs hexadécimales pour le chargeur BASIC: 3E,nouvelle adresse de la banque,CD,06,BC,C9,0

Respectez bien les indications que nous vous donnons à propos des registres de transmission lors du choix des paramètres.

Format assembleur:

```
ld a,N
call 06,BC
ret
```

### 6.6.3 ATTENDRE LA FRAPPE D'UNE TOUCHE

Adresse : BB06 (hex)

48878 (déc)

Cette routine interrompt le déroulement du programme juqu'à ce qu'une touche quelconque soit frappée.

Registres de transmission: - - -

Registres de reponse: a l'accumulateur contient le code de la touche enfoncée.

Valeurs hexadecimales pour le chargeur BASIC: - - -

Cette fonction peut être appelée en n'importe quel endroit de votre programme simplement par l'instruction CALL &BB06.

Format assembleur: call 06,BB

## 6.6.4 SCROLLING

Adresse : BC4D (hex)
48205 (déc)

Cette routine vous permet de déplacer la totalité de l'écran d'une ligne vers le haut ou le bas. La valeur de la couleur de la nouvelle ligne (PAPER) peut être choisie librement, dans le cadre des possibilités du mode écran choisi.

Registres de transmission: b le registre b indique la direction du décalage. Un 0 indique un scrolling descendant, toute autre valeur un scrolling ascendant.
a l'accumulateur indique le code couleur pour la nouvelle ligne créée.

Registres de réponse: - - -

Valeurs hexadécimales pour le chargeur BASIC:
scrolling descendant: 6,0,3E,code couleur,CD,4D,BC,C9,0
scrolling ascendant: 6,1,3E,code couleur,CD,4D,BC,C9,0

Format assembleur:

```
ld b,N
ld a,N
call 4D,BC
ret
```

### 6.6.5 SCROLLING DE ZONES DE L'ECRAN

Adresse : BC50 (hex)
48208 (déc)

Cette routine vous permet de décaler des zones partielles de l'écran (fenêtres).

Registres de transmission:

- b le registre b indique la direction du décalage. Un 0 indique un scrolling descendant, toute autre valeur un scrolling ascendant.
- a l'accumulateur indique le code couleur pour la nouvelle ligne créée.
- h colonne limite gauche de la fenêtre
- d colonne limite droite de la fenêtre
- l ligne limite supérieure de la fenêtre
- e ligne limite inférieure de la fenêtre

Faites très attention à ne pas fournir comme valeurs limites des valeurs situées en dehors de l'écran car celà pourrait bien "planter" votre CPC 464. L'angle supérieur gauche de l'écran a la valeur 0,0.

Registres de réponse: - - -

Valeurs hexadècimales pour le chargeur BASIC:

6, direction, 3E, code couleur, 21, limite supérieure, limite gauche, 11, limite inférieure, limite droite, CD, 50, BC, C9, 0

Les limites ainsi définies de la zone à "scroller" ne sont en aucun cas dèpendantes des fenêtres que vous aviez définies auparavant.

Format assembleur:

```
ld b,N
ld a,N
ld hl,NN
ld de,NN
call 50,BC
ret
```

## 6.6.6 SELECTION DU MODE ECRAN

Adresse : BC0E (hex)

48142 (déc)

Cette routine vous permet de changer de mode écran en langage-machine.

Registres de transmission: a l'accumulateur contient la valeur du nouveau mode sélectionné.

La valeur de l'accumulateur doit bien sûr être entre 0 et 2.

Registres de réponse: - - -

Valeurs hexadécimales pour le chargeur BASIC: 3E,nouveau mode,CD,E,BC,C9,0

Format assembleur:

```
ld a,N
call 0E,BC
ret
```

### 6.6.7 INVERSION VIDEO D'UN CARACTERE

Adresse : BC4A (hex)

48202 (déc)

Cette routine vous permet d'inverser un caractère défini par ses coordonnées (0,0 = angle supérieur gauche de l'écran). L'inversion se produit sur deux cartes bits grâce à un double EXCLUSIVE OR.

Registres de transmission: b carte bits d'inversion 1 (couleur 1)

c carte bits d'inversion 2 (couleur 2)

h colonne

l ligne

Les paramètres dépendent bien sûr du mode choisi.

Registres de réponse: - - -

Valeurs hexadécimales pour le chargeur BASIC: 6,carte bits 1,E,carte bits 2,21,colonne,ligne,CD,4A,BC,C9,0

Exemple: mode 1, couleur du caractère jaune, couleur du fond bleu. Le caractère à inverser se trouvant dans les coordonnées fournies est un 0. Examinons l'inversion des deux lignes supérieures de la carte bits de ce caractère. Comme carte bits d'inversion 1, nous choisissons "1000 0000" et comme carte bits d'inversion 2, "0000 1111".

| | | |
|---|---|---|
| Situation de départ | 0111 0000 | 1100 0000 |
| Après l'inversion 1 | 1111 0000 | 0100 0000 |

| | |
|---|---|
| Après l'inversion 2 | 1111 1111 0100 1111 |
| Couleur du pixel | RRRR RRRR BBRR BBBB |

R=rouge B=bleu

Format assembleur:

```
ld b,N
ld c,N
ld hl,NN
call 4A,BC
ret
```

### 6.6.8 SCROLLING HORIZONTAL DE L'ECRAN

Adresse : BC05 (hex)
48233 (déc)

Cette routine vous permet de décaler la totalité de l'écran vers la gauche ou vers la droite. La colonne qui disparaît à gauche réapparaît à droite (wrap around).

Registres de transmission: h octet fort de la valeur de décalage
l octet faible de la valeur de décalage

La valeur de décalage doit absolument être un nombre pair.

Registres de réponse: - - -

Valeurs hexadécimales pour le chargeur BASIC: 21, octet faible de la valeur de décalage, octet fort de la valeur de décalage, CD, 5, BC, C9, 0

Format assembleur:

```
ld hl,NN
call 05,BC
ret
```

6.6.9 POSITIONNEMENT DU CURSEUR

Adresse : BB75 (hex)

47989 (déc)

Cette routine correspond à l'instruction BASIC LOCATE.

Registres de transmission: h nouvelle colonne du curseur

l nouvelle ligne du curseur

La routine ne vérifie pas si les valeurs fournies restent à l'intérieur de l'écran.

Registres de réponse: - - -

Valeurs hexadécimales pour le chargeur BASIC: 21,nouvelle ligne,nouvelle colonne,CD,75,BB,C9,0

Format assembleur:

```
ld hl,NN
call 75,BB
ret
```

6.6.10 POSITIONNEMENT HORIZONTAL DU CURSEUR

Adresse : BB6F (hex)

47983 (déc)

Cette routine vous permet de positionner le curseur à l'intérieur de la ligne sur laquelle il se trouve.

Registres de transmission: a nouvelle colonne du curseur

La routine ne vérifie pas si la valeur fournie reste à l'intérieur de l'écran.

Registres de réponse: - - -

Valeurs hexadécimales pour le chargeur BASIC: 3E,nouvelle colonne,CD,6F,BB,C9,0

Format assembleur:

```
ld a,N
call 6F,BB
ret
```

6.6.11 POSITIONNEMENT VERTICAL DU CURSEUR

Adresse : BB72 (hex)

47986 (déc)

Cette routine vous permet de positionner le curseur à l'intérieur de la colonne sur laquelle il se trouve.

Registres de transmission: a nouvelle ligne du curseur

La routine ne vérifie pas si la valeur fournie reste à l'intérieur de l'écran.

Registres de réponse: - - -

Valeurs hexadécimales pour le chargeur BASIC: 3E,nouvelle ligne,CD,72,BB,C9,0

Format assembleur:

```
ld a,N
call 72,BB
ret
```

6.6.12 INTERROGATION DU JOYSTICK (MANCHE A BALAI)

Adresse : BB24 (hex)
47908 (déc)

Cette routine permet de lire les valeurs des commutateurs des deux joysticks que vous pouvez connecter à votre ordinateur.

Registres de transmission: - - -

Registres de réponse:
a l'accumulateur contient après l'interrogation le code correspondant à la position du joystick
h idem
l idem mais pour le joystick numéro 2

Tout bit allumé correspond à un contact. Les différents bits correspondent aux valeurs suivantes:

bit 0 haut
bit 1 bas
bit 2 gauche
bit 3 droite
bit 4 fire 2 (bouton fire standard)
bit 5 fire 1
bit 6 inutilisé
bit 7 inutilisé

Valeurs hexadécimales du chargeur BASIC: CD,24,BB,32,octet faible

fire 2,octet fort fire 2,7D,32,octet faible fire 1,octet fort fire 1,C9,0

Après avoir appelé cette routine vous pouvez lire l'état des deux joysticks grâce à deux simples PEEK.

Format assembleur:

```
call 24,BB
ld (NN),a
ld a,l
ld (NN),a
ret
```

### 6.6.13 BRANCHER ET DEBRANCHER L'INVERSION

Adresse : BB9C (hex)

48028 (déc)

Cette routine vous permet d'échanger les couleurs actuelles de PAPER et PEN. Un premier appel de cette routine réalise l'inversion, un second appel la supprime. Vous pouvez arriver au même résultat en BASIC avec PRINT CHR$(24).

Registres de transmission: - - -

Registres de réponse: - - -

Vous pouvez appeler cette routine simplement par
CALL &BB9C

Format assembleur: call 9C,BB

## 6.7 RANDOMIZE - A LA POURSUITE DU HASARD

Votre CPC 464 dispose de deux instructions vous permettant de recréer le hasard. La première est la fonction RND qui va chercher, chaque fois qu'elle est appelée la prochaine valeur d'une série de nombres aléatoires. Essayez l'exemple suivant après avoir déclenché un RESET général de votre ordinateur (CTRL-SHIFT-ESC):

```
10 REM Hasard?
20 FOR t=1 TO 5
30 PRINT RND
40 NEXT t
```

Si vous comparez les chiffres obtenus avec la liste ci-dessous, vous constaterez qu'il n'y a aucune différence:

```
0.271940658
0.528612386
0.021330127
0.175138616
0.657773343
```

Vous voyez que la fonction aléatoire n'est qu'une fonction pseudo-aléatoire: la série de chiffres à laquelle elle fait appel, pour être arbitraire n'en est pas moins rigoureusement invariable et la fonction RND la lit du début après chaque initialisation de l'ordinateur. Si donc vous n'utilisiez dans vos jeux que cette seule fonction pour générer un soi-disant hasard, vous pourriez en assez peu de temps prévoir l'enchaînement des divers évènements

aléatoires censés agrémenter votre jeu. A la longue celà deviendrait très ennuyeux.
C'est pourquoi il existe une autre instruction: RANDOMIZE xx où xx est un paramètre vous permettant de faire commencer la lecture par RND du prochain nombre de la série aléatoire en un endroit calculé en fonction de ce paramètre.

Vous pouvez recommencer notre petite expérience avec ces nouvelles données: provoquez un RESET de l'ordinateur puis modifiez le programme Hasard? de tout à l'heure en y ajoutant la ligne:

```
15 RANDOMIZE 33
```

Lancez cette nouvelle version avec RUN et notez les chiffres auxquels vous aboutissez. Vous constatez un premier progrès: les chiffres obtenus sont différents de la liste que nous donnions plus haut. Donc si l'on ne connaît pas le paramètre de RANDOMIZE, on ne peut pas savoir par avance quels seront les chiffres tirés "au hasard".

Ce progrès reste cependant très mince car vous pouvez constater que si vous provoquez à nouveau un RESET et que vous lancez à nouveau la nouvelle version du programme Hasard? avec RANDOMIZE 33 vous aboutirez aux mêmes chiffres que vous venez de noter. Pour recréer véritablement une situation de hasard, il faut faire appel à des facteurs qui ne puissent être appréhendés directement par un être humain. Le plus simple est d'utiliser l'horloge de l'ordinateur. L'horloge marque le temps qui s'est écoulé depuis l'allumage de votre ordinateur. Ce temps est mesuré en unités de 3/100 secondes. Il est donc possible d'affecter à l'instruction

RANDOMIZE un paramètre dépendant du temps qui s'est écoulé depuis la mise sous tension de votre ordinateur, temps qui constitue, surtout calculé de manière aussi précise, un facteur que vous ne pouvez connaître. Voici comment nous pouvons maintenant, à l'aide de l'horloge de l'ordinateur et de ses deux instructions pseudo-aléatoires, recréer le hasard:

```
10 REM Hasard
15 RANDOMIZE TIME
20 FOR t=1 TO 5
30 PRINT RND
40 NEXT t
```

La ligne essentielle de cette nouvelle version de notre programme Hasard est la ligne 20 qui utilise la fonction RANDOMIZE avec la variable systeme TIME qui contient le temps de l'horloge de votre ordinateur.

Les nombres fournis par la fonction RND sont toujours supérieurs à 0 et inférieurs à 1. Pour obtenir des nombres entiers dans le cadre qui vous intéresse, il faut que vous appliquiez aux nombres renvoyés par la fonction RND la formule suivante:

nombre aléatoire = INT (RND * (limite supérieure - limite inférieure + 1)) +limite inférieure.

Par exemple si vous voulez un nombre compris entre 2 et 12:

Vous calculez d'abord le nombre de possibilités: 12 (limite supérieure) - 2 (limite inférieure) + 1 = 11.

Il y bien en effet 11 nombres compris entre 2 et 12 inclus. Vous multipliez donc 11 (nombre de possibilités) par RND (nombre pseudo-aléatoire) et vous obtenez un nombre entre 0 et 10 que vous arrondissez grâce à la fonction INT. Vous ajoutez ensuite 2 (limite inférieure) à ce nombre et vous obtenez un nombre parfaitement aléatoire compris entre 2 et 12.

## 6.8 LE CPC COMME CALCULATRICE

Le seul but recherché par les constructeurs des premiers ordinateurs était de réaliser des machines qui puissent effectuer les calculs les plus compliqués beaucoup plus rapidement que les hommes. Bien que tout ordinateur garde toujours la faculté de remplir cette fonction, cette caractéristique est toutefois secondaire, en ce qui concerne les ordinateurs domestiques, par rapport aux autres possibilités de ces ordinateurs. Nous allons malgré tout, dans cette section, essayer de donner quelques idées à ceux qui utilisent le CPC 464 avant tout pour effectuer des calculs.

Nous allons tout d'abord vous montrer comment transformer votre ordinateur en une "calculatrice de poche", si ce n'est par la taille, du moins par la facilité d'utilisation. Nous allons simplement redéfinir à cet effet le clavier de façon à ce qu'aucun calcul ne nécessite l'emploi de la touche SHIFT (pour les opérateurs) et que d'une manière générale tout calcul puisse s'effectuer sans utiliser aucune combinaison de deux touches différentes.

Si celà ne vous gêne pas de ne pouvoir utiliser que les chiffres situés au-dessus des lettres, vous pouvez également taper les lignes 200 et suivantes de notre programme qui affectent au bloc de chiffres les fonctions (SIN(X);COS(X) que vous trouvez sur une calculatrice normale.

L'utilisation du programme est très simple. Dès que vous l'avez lancé, vous voyez apparaître le message "MODIFICATIONS EFFECTUEES"

qui indique que toutes les touches comportant un opérateur arithmétique peuvent être utilisées sans SHIFT. D'autre part la touche P s'est vu affecter le ? ce qui vous permet d'entrer plus rapidement l'instruction PRINT.

Voici une table de ces modifications:

| Signes mathématiques | Touche | Valeur de la touche |
|---|---|---|
| * | : | 29 |
| + | ; | 28 |
| ( | | 17 |
| ) | | 19 |
| & | | 26 |
| SIN( | 0 | 128 |
| COS( | 1 | 129 |
| TAN( | 2 | 130 |
| ATN( | 3 | 131 |
| ABS( | 4 | 132 |
| INT( | 5 | 133 |
| HEX$( | 6 | 134 |
| BIN$( | 7 | 135 |
| STR$( | 8 | 136 |
| ? | P | 27 |

Si vous voulez vous servir régulièrement de ce programme, nous vous conseillons de coller sur les touches modifiées de petites étiquettes indiquant leur nouvelle fonction de façon à ce que vous n'ayez pas à les chercher sans cesse.

```
10 REM calculatrice de poche
20 MODE 2
30 KEY DEF 29,1,42
40 KEY DEF 28,1,43
50 KEY DEF 17,1,40
60 KEY DEF 19,1,41
70 KEY DEF 26,1,38
80 KEY DEF 27,1,63
90 GOSUB 200
100 PRINT"MODIFICATIONS EFFECTUEES"
110 END

200 KEY 128,"sin("
210 KEY 129,"cos("
220 KEY 130,"tan("
230 KEY 131,"atn("
240 KEY 132,"abs("
250 KEY 133,"int("
260 KEY 134,"hex$("
270 KEY 135,"bin$("
280 KEY 136,"str$("
290 RETURN
```

Commentaire du listing:

----------------------

| | |
|---|---|
| 10 | Titre |
| 20 | Passage en mode 80 caractères |
| 30-80 | KEY DEF permet de modifier l'ordre des touches comportant des opèrateurs mathématiques |

| | |
|---|---|
| 90 | Saut à la sous-routine 200. Vous pouvez supprimer cette ligne si vous ne voulez pas de la sous-routine 200. |
| 100 | Edition du message |
| 110 | Fin du programme |
| 200-280 | Redéfinition des touches du bloc des chiffres |
| 290 | Retour au programme principal |

## 6.8.1 PRECISION DE CALCUL

La précision de calcul d'un ordinateur est fonction du nombre de chiffres après la virgule qui correspondent aux chiffres du nombre réel. Un ordinateur qui renvoie 3.1416 pour PI n'a pas une grande précision de calcul.
Votre CPC ne peut stocker correctement "que" neuf chiffres avant ou après la virgule, ce qui signifie que pour des nombres comportant plus de chiffres, l'imprécision sera supérieure à 1.

Un exemple; entrez: PRINT 5E9+1-5E9

Vous attendiez certainement que votre ordinateur réponde 1 mais vous voyez qu'il a répondu 2. Il y a donc une imprécision de 1.

Le plus grand nombre qui puisse être stocké correctement est:
2↑32-1 = 4294967295

Cette relative imprécision tient au mode de stockage des variables que nous avons décrit au chapitre 5.

Pour augmenter la précision de calcul, vous pouvez utiliser des formules à la place des valeurs dont vous avez besoin. Il faut cependant faire attention à ne pas employer de constantes telles que les racines ou π car celles-ci ne font que nuire à la précision de calcul.

Nous avons créé un programme à partir d'une telle formule pour vous montrer comment vous pouvez introduire vos formules propres dans vos programmes sans utiliser la fonction DEF FN qui ne peut

pas être employée dans tous les cas. Notre formule calcule EXP (X) de la façon suivante:

EXP(X)=1+X/1!+X↑2/2!+X↑3/3!

Notre programme vous permet d'élever la ligne supérieure jusqu'au facteur 31. Ensuite on atteint le plafond 1E34.
Cette fonction que ne possède pas votre CPC est simulée par la routine 100.
Pour utiliser le programme, il suffit d'entrer la valeur X voulue et vous voyez s'afficher après un court temps de calcul la valeur EXP(X).

```
10 REM EXP(x)
20 INPUT"x-VALEUR";x
30 FOR n=1 TO 31
40 x1=x↑n
50 GOSUB 100
60 erg1=x1/fa
70 erg2=erg2+erg2+erg1
80 NEXT
90 PRINT erg2+1
95 END
100 fa=1
110 FOR m=1 TO n
120 fa=fa*m
130 NEXT m
140 RETURN
```

Commentaire du programme:

-------------------------

10 titre

20 entrée de la valeur X

30 ouverture de la boucle qui définit le nombre de facultés

40 constitution des puissances de X

50 saut a la sous-routine 100

60 traitement des facultés

70 addition du résultat

80 fermeture de la boucle

90 edition du résultat

95 fin du programme

100 rétablissement de la faculté

110 ouverture de la boucle pour la faculté

120 formule de calcul de la faculté

130 fermeture de la boucle

140 retour au programme principal

Une formule telle que celle de EXP(X) peut difficilement être définie avec une instruction DEF FN car les 31 parties ne peuvent entrer dans l'instruction DEF FN.

Un autre exemple d'application des formules est le calcul des points 0. Les points 0 sont les points de rencontre entre la courbe d'une fonction et l'axe des X. Ce sont des points très importants pour l'analyse de la courbe d'une fonction qui ne peuvent être dégagés que très imprécisément du seul dessin.

Voici un exemple de calcul de points 0 avec une équation carrée:

F(X)=X↑2+P*X+Q

La formule de calcul des points 0 est:

X1= -P/2+SQR(P↑2/4*Q); X2= -P/2-SQR(P↑2/4*Q)

Vous devez entrer d'abord la valeur de P, puis celle de Q et le programme vous donnera la valeur des points 0 (X1,X2).

```
10 INPUT p
20 INPUT q
30 x1= -p/2+SQR(p*p/4-q)
40 x2= -p/2-SQR(p*p/4-q)
50 IF x1*x2<>q THEN 80
60 PRINT x1,x2
70 END
80 PRINT"Pas de points zeros"
90 END
```

Commentaire du listing:
-----------------------

| | |
|---|---|
| 10 | entrée de p |
| 20 | entrée de q |
| 30 | calcul du premier point 0 |
| 40 | calcul du second point 0 |
| 50 | vérification des points 0 d'après le théorème de Vieta |
| 60 | édition des points 0 |
| 70 | fin du programme |
| 80 | affiche qu'il n'y a pas de points 0 |

Si le programme est interrompu par un message d'erreur, c'est qu'il n'y a pas de points 0.

### 6.8.2 VITESSE DE CALCUL

La vitesse de calcul constitue une caractéristique importante d'un ordinateur. Elle permet à l'utilisateur d'évaluer correctement les possibilités de son ordinateur.

On calcule généralement la vitesse de calcul grâce à un programme de test que l'on fait tourner sur différents ordinateurs. On peut ainsi effectuer des comparaisons valables et voir ainsi quel ordinateur convient le mieux à une application donnée.

Les tâches standard qu'on fait exécuter par les ordinateurs pour calculer leur vitesse portent le nom générique de tests BENCHMARK. Nous avons utilisé un tel test dans ce chapitre en suivant une méthode très répandue, "le tamis d'Eratosthène".

La vitesse de calcul comprend également le traitement des interrogations, des boucles et la gestion de champs de données. Ces tâches sont également prises en compte par la méthode que nous avons choisie et que nous avons appliquée à deux ordinateurs avec les résultats suivants:

| | |
|---|---|
| Apple II Europlus avec MBASIC 5.2: | 391,8 secondes |
| VIC 20 avec carte 64K: | 394,8 secondes |

Pour que vous puissiez vous même tester combien de temps il faut à votre CPC pour exécuter ce test, voici un listing du programme:

```
10 REM Eratosthène
20 DEFINT a-z
```

```
30 DIM fl(1000):l=1000:PRINT"Début"
40 FOR j=1 TO 10
50 co=0
60 FOR i=1 TO l:fl(i)=1:NEXT i
70 FOR i=1 TO l
80 IF fl(i)<>1 THEN 160
90 prim=i+i+3
100 k=i+prim
110 IF k>l THEN 150
120 fl(k)=0
130 k=k+prim
140 GOTO 110
150 co=co+1
160 NEXT i
170 NEXT j
180 PRINT co;"Nombres premiers"
190 END
```

Si vous lancez ce programme et mesurez le temps que met votre ordinateur, vous verrez qu'il soutient la comparaison avec les autres ordinateurs, y compris en ce qui concerne la vitesse de calcul.

Un dernier conseil: votre CPC possède un bon nombre de signes mathématiques spéciaux que vous ne pouvez faire afficher qu'avec une instruction PRINT CHR$(X). Mais en réfléchissant un peu au problème vous devriez pouvoir trouver un moyen de faire jouer à ces signes leur rôle normal. Essayez par exemple avec la constante PI.

## 6.9 MOUVEMENTS DE L'ECRAN

Plusieurs dialectes BASIC possèdent une instruction SCROLL permettant de faire glisser le contenu de l'écran d'une ligne vers le bas ou le haut. Nous avons utilisé une routine de la ROM du CPC pour simuler cette instruction.

L'instruction SCROLL peut être très utile soit pour des jeux, soit pour faire défiler une liste.

La méthode la plus simple pour provoquer un mouvement de l'écran consiste à ajouter un caractère dans l'angle inférieur droit, ce qui provoque un scrolling vertical vers le haut. Mais il est beaucoup plus compliqué d'arriver à un scrolling vertical vers le bas. On utilise généralement pour celà des routines en langage-machine assez longues. Sur le CPC cependant il est possible d'utiliser des routines de la ROM.

La routine de mouvement de l'écran se trouve en &BC4D (voyez le chapitre 6.6). Elle déplace l'écran de 8 pixels (un caractère). La direction dépend de la valeur du registre b. Si celle-ci est nulle le déplacement se fait vers le bas et si celle-ci est non nulle le déplacement se fait vers le haut.

Il y a deux façon d'appeler cette routine: on peut écrire une routine d'appel en langage-machine dans laquelle on pokera en BASIC la valeur à affecter au registre b, ou on peut écrire dès le départ deux routines différentes, l'une pour un mouvement descendant, l'autre pour un mouvement ascendant.

Pour plus de commodité, nous avons choisi cette dernière méthode. Voyons donc ce que devra faire notre routine en langage-machine:

- tout d'abord charger une valeur donnée dans le registre b avec l'instruction ld b,N qui a le code 06. Le paramètre utilisé avec cette instruction sera 0 pour un mouvement descendant et 255 pour un mouvement ascendant. C'est d'ailleurs la seule différence entre les deux routines.

- il faut ensuite appeler la routine de la ROM avec l'instruction CALL NN dont le code est 205. A la suite de l'instruction doit figurer l'adresse &BC4D divisée en octets faible et fort: 77 (4D) et 188 (BC). Il faut enfin retourner au BASIC avec un ret dont le code est 201.

Voici les deux routines en langage-machine:

Pour le mouvement ascendant:

```
ld b,ff         06,255
call &BC4D      205,77,188
ret             201
```

Pour le mouvement descendant:

```
ld b,0          06,0
call &BC4D      205,77,188
ret             201
```

Le programme BASIC de chargement de ces routines les place en

43880 pour le mouvement ascendant et en 43886 pour le mouvement descendant. Ce programme une fois exécuté, vous pouvez l'effacer avec l'instruction DELETE -140 pour écrire ensuite votre propre programme qui utilisera ces routines.

```
10 REM ascendant
20 MEMORY 43879
30 DATA 06,255,205,77,188,201
40 FOR adr=43880 TO 43885
50 READ valeur
60 POKE adr,valeur
70 NEXT
80 REM descendant
90 DATA 06,0,205,77,188,201
100 FOR adr=43886 TO 43891
110 READ valeur
120 POKE adr,valeur
130 NEXT adr
140 END
```

Commentaire du listing:

-----------------------

| | |
|---|---|
| 10 | titre |
| 20 | réduire la mémoire BASIC |
| 30 | DATAs pour la routine ascendante |
| 40 | ouverture d'une boucle d'écriture de la routine en langage-machine |
| 50 | lecture des valeurs |
| 60 | écriture des valeurs dans les adresses correspondantes |
| 70 | fermeture de la boucle |

80 titre

90 DATAs pour la routine descendante

100 ouverture d'une boucle d'écriture de la routine en langage-machine

110 lecture des valeurs

120 écriture des valeurs dans les adresses correspondantes

130 fermeture de la boucle

140 fin du programme

Ces routines une fois chargées, vous pourrez les appeler simplement avec call 43880 pour le mouvement ascendant et 43886 pour le mouvement descendant.

Voici maintenant une démonstration des possibilités de scrolling que vous offrent ces deux routines: nous allons placer une inscription sur l'écran pour la faire ensuite monter puis descendre de 5 lignes grâce à nos deux routines.

Vous pouvez contrôler la vitesse du mouvement en appuyant sur une touche. Si vous appuyez sur une touche, l'inscription se déplace. Sinon, le programme attend que vous appuyiez sur une touche.

```
10 REM scroll demo
20 MODE 1
30 LOCATE 15,12:PRINT"SCROLLDEMO"
35 REM ascendant
40 FOR n=1 TO 5
50 IF INKEY$="" THEN 50
60 CALL 43880
70 NEXT n
```

```
75 REM descendant
80 FOR n=1 TO 5
90 IF INKEY$="" THEN 90
100 CALL 43886
110 NEXT n
120 END
```

Commentaire du listing:

------------------------

| | |
|---|---|
| 10 | titre |
| 20 | fixation du mode écran |
| 30 | édition de l'inscription |
| 35 | titre |
| 40 | ouverture de la boucle pour compter les mouvements |
| 50 | interrogation du clavier |
| 60 | appel de la routine scroll (ascendante) |
| 70 | fermeture de la boucle |
| 75 | titre |
| 80 | ouverture de la boucle pour compter les mouvements |
| 90 | interrogation du clavier |
| 100 | appel de la routine scroll (descendante) |
| 110 | fermeture de la boucle |
| 120 | fin du programme |

N'oubliez pas que ce programme de démonstration ne peut fonctionner si vous n'avez pas fait tourner au préalable le programme de création des routines de scrolling.

Pour faciliter l'appel de ces routines vous pouvez définir deux variables correspondant aux deux adresses:

HAUT=43880        BAS=43886

Il est intéressant de ne pas perdre de vue que l'accumulateur contient, pour ces deux routines, le code couleur de la ligne ajoutée en bas ou en haut de l'écran du fait du scrolling. Ceci vous permet de créer encore d'autres fonctions grâce à la routine scroll.

Nous avons ainsi écrit un petit programme qui grâce à la routine scroll vous permet de modifier la direction, la longueur du mouvement ainsi que la couleur de la ligne nouvelle.

Pour obtenir ces fonctions supplémentaires, il faut ajouter quelques instructions à notre routine en langage-machine. Le registre b sera toujours utilisé pour indiquer la direction du mouvement mais il fonctionnera en outre comme compteur du nombre de lignes à ajouter. Le registre a contiendra le code couleur de la ligne nouvelle.

Comme le contenu des registres est détruit lors de l'appel d'une routine de la ROM, il nous faut placer le contenu de ces registres sur la pile où nous irons les rechercher après l'appel de la routine. Pour ce faire nous utiliserons les instructions PUSH (pousser) et POP (retirer).

Notre routine ainsi étendue devient donc:

| adresse | instruction | code | explication |
| --- | --- | --- | --- |
| 43870 | ld b,N | 06 | le nombre de lignes (N) est chargé |
| 43871 | | N | dans le registre b |
| 43872 | ld a,M | 62 | le code couleur (M) est chargé |
| 43873 | | M | dans l'accumulateur |
| 43874 | push bc | 197 | b est placé sur la pile |
| 43875 | push af | 245 | a est placé sur la pile |
| 43876 | ld b,r | 06 | le code direction (R) est chargé |
| 43877 | | R | dans le registre b |
| 43878 | call &BC4D | 205 | appel de la routine de la ROM |
| 43879 | | 77 | octet faible de l'adresse |
| 43880 | | 188 | octet fort de l'adresse |
| 43881 | pop af | 241 | a est retiré de la pile |
| 43882 | pop bc | 193 | b est retiré de la pile |
| 43883 | djnz,e | 16 | voir plus loin |
| 43884 | | 245 | |
| 43885 | ret | 201 | retour au BASIC |

L'instruction djnz,e est une instruction complexe de branchement conditionnel et relatif: elle signifie "décrementer, sauter de e si non nul.

Le déroulement du programme exige en effet qu'il y ait un saut à l'adresse 43874, de façon à ce que le nombre de lignes ne soit pas limité à 1. Le registre b, comme nous l'avons déjà indiqué sert aux opérations de comptage. Ce registre doit donc être décrémenté de 1 chaque fois que la routine est exécutée. C'est ce que fait notre nouvelle instruction. En outre, si le registre b n'est toujours pas nul après cette décrémentation, le programme saute à

nouveau à l'adresse 43874.

La valeur 245 qui sert de paramètre à notre nouvelle instruction indique au programme qu'il doit sauter de 9 octets en arrière. A ces 9 octets, deux sont ajoutés du fait que le compteur de programme est déjà dirigé sur l'instruction suivante. C'est ainsi que nous obtenons le nombre 11. Pour indiquer qu'il s'agit d'un saut en arrière, et non en avant, cette valeur doit être retranchée de 256 et on arrive alors à notre valeur de 245.

Si le registre b vaut 0, le programme retourne au BASIC.

```
10 REM scrolldemo2
20 MEMORY 43869
30 adr=43869:scroll=43870
40 DATA 6,10,62,0,197,245,6,255,205,77
50 DATA 188,241,193,16,245,201
60 FOR co=1 TO 16
70 READ valeur
80 POKE adr+co,valeur
90 NEXT co
100 MODE 1
110 INPUT"Couleur";m
120 INPUT"nombre de lignes";n
130 INPUT"vers le haut (1) ou le bas (2)";r
140 IF r=1 THEN r=255 ELSE r=0
150 POKE 43873,m
160 POKE 43871,n
170 POKE 43877,r
180 CALL scroll
```

```
190 GOTO 100
```

Commentaire du listing:

----------------------

10 titre

20 limitation de la mémoire BASIC

30 fixation de l'adresse de début du programme

40-50 Datas pour la routine en langage-machine

60 ouverture de la boucle de lecture des DATA

70 lecture des codes langage-machine

80 écriture des valeurs dans les adresses correspondantes

90 fermeture de la boucle

100 fixation du mode écran

110 entrée du code couleur

120 entrée du nombre de lignes

130 décision si mouvement ascendant ou descendant

140 traitement de la dernière entrée

150 écriture du code couleur

160 écriture du nombre de lignes

170 écriture de la direction du mouvement

180 appel de la routine en langage-machine

190 retour en 100

Les entrées nécessaires dans ce programme doivent rester dans les limites suivantes:

code couleur: entre 0 et 255

nombre de lignes: entre 1 et 25

Si vous voulez intégrer la routine en langage-machine dans un de vos programmes, vous aurez besoin des adresses suivantes:

43873 le code couleur doit être placé dans cette adresse

43871 le nombre de lignes doit être placé dans cette adresse

43877 la direction du mouvement doit être placée dans cette adresse

43870 adresse de départ de la routine en langage-machine

## 6.10 TRI DE DONNEES

Le tri de données, qu'il s'agisse de données numériques ou de textes est une des tâches les plus classiques des programmeurs. Le critère de tri est naturellement la grandeur même des nombres lorsqu'il s'agit de données numériques et l'ordre alphabétique lorsqu'il s'agit de textes. Les codes des lettres de l'alphabet ont été choisis de façon à rendre très simple un tri par ordre alphabétique. Notez d'ailleurs que vous n'avez pas à vous en préoccuper car l'interpréteur BASIC vous permet de comparer directement deux chaînes de caractères entre elles d'après l'ordre alphabétique.

Pour trier des données on peut faire appel à quantité d'algorithmes bien connus qui se différencient les uns des autres d'une part par leur complexité et d'autre part par leur efficacité. Pour choisir le bon algorithme pour une application donnée il convient tout d'abord de tenir compte du nombre de données qu'il s'agit de trier. Si vous n'avez en effet qu'une dizaine ou une cinquantaine de données à trier, les procédures les plus simples suffiront en général aisément à la tâche. Par contre si la masse de données excède une centaine ou un millier, les procédures les plus simples se révèleront le plus souvent impraticables car le temps de tri risque de dépasser plusieurs heures. Pour que vous puissiez aprécier la signification du choix d'un algorithme de tri, nous allons vous présenter deux méthodes de tri, l'une élémentaire et l'autre très rapide, bubble sort et quick sort.

Avec bubble sort on compare deux éléments voisins qu'on échange si

nécessaire. Cet algorithme peut être programmé en BASIC avec deux boucles imbriquées:

```
100 FOR i=1 TO n:fl=0
110 FOR j=n TOi step-1
120 IF a$(j-1)>a$(j) THEN h$=a$(j):a$(j)=a$(j-1):a$(j-1)=h$:fl=1
130 NEXT j:IF fl=0 THEN RETURN
140 NEXT i:RETURN
```

Ce programme fonctionne comme un sous-programme que vous pouvez appeler avec GOSUB 100. La dimension du champ (l'index maximum) doit être transmise à la variable n, par exemple:

```
10 n=50:DIM a$(n)
20 'lecture des données
30 GOSUB 100 'tri de données
```

Un tel programme de tri est parfaitement adapté à une situation où il n'y a que peu de données à trier ou bien où les données sont déjà prétriées et où seul un nouvel enregistrement doit être inséré dans les données triées.

Voici maintenant un programme de tri connu sous le nom de quicksort qui convient au contraire au tri de masses importantes de données. Il s'agit ici aussi de trier d'après un champ alphanumérique dont l'index limite doit être placé dans la variable n.

Ce programme génère comme démonstration le nombre que vous choisissez de mots aléatoires qu'il affiche, qu'il trie puis

affiche dans l'ordre résultant du tri. Le temps qui a été nécessaire pour effectuer le tri est affiché au même moment.

```
10 DEFINT b-z
20 INPUT "Nombre de mots  ";n: DIM d$(n),o(20),u(20)
30 FOR i=1 TO n: b=5+10*RND
40 FOR j=1 TO b: d$(i)=d$(i)+CHR$(RND*25+65):NEXT: NEXT
50 GOSUB 80: a=TIME: GOSUB 100: a=(TIME-a)/300
60 PRINT: GOSUB 80: PRINT: PRINT a "Sek.": END
70 '
80 FOR i=1 TO n: PRINT d$(i): NEXT: RETURN
90 '
100 s=1: o(1)=1: u(1)=n
110 l=o(s): r=u(s): s=s-1
120 i=l: j=r: h$=d$((l+r)/2)
130 WHILE d$(i)<h$ AND i<r: i=i+1: WEND
140 WHILE d$(j)>h$ AND j>l: j=j-1: WEND
150 IF i<=j THEN d$=d$(i):d$(i)=d$(j):d$(j)=d$:i=i+1:j=j-1
160 IF i<=j THEN 130
170 IF r-i<=j-l THEN 200
180 IF l<j THEN s=s+1: o(s)=l: u(s)=j
190 l=i: GOTO 220
200 IF i<r THEN s=s+1: o(s)=i: u(s)=r
210 r=j
220 IF r>l THEN 120
230 IF s>0 THEN 110
240 RETURN
```

Si vous faites tourner le programme avec des champs de différentes tailles, vous obtiendrez environ les résultats suivants:

| n | sec. |
|---|---|
| 50 | 3.8 |
| 100 | 7.9 |
| 200 | 18.6 |
| 500 | 110 |

Le fait que le temps de tri pour 500 enregistrements soit proportionnellement beaucoup plus long que pour les masses de données plus réduites tient à un autre phénomène, la garbage collection. Ce phénomène devient inévitable lorsque vous travaillez avec un grand nombre de chaînes de caractères. Si vous venez juste d'exécuter le programme avec 500 chaînes de caractères et que vous essayez d'entrer:

```
? FRE("")
```

vous devrez attendre environ 29 secondes avant que l'ordinateur ne vous réponde. C'est le temps qu'il faut à l'interpréteur BASIC pour ranger sa mémoire de chaînes de caractères. Chaque fois que le contenu d'une chaîne de caractères est modifié, l'interpréteur écrit le nouveau contenu dans la mémoire disponible et laisse intact l'ancien contenu. Mais lorsque toute la mémoire disponible pour les chaînes de caractères est occupée entièrement, l'ordinateur examine toutes les chaînes et supprime celles qui sont périmées de façon à libérer à nouveau de la place pour les nouvelles chaînes à créer. Plus il y a de chaînes valables en mémoire, plus cette procédure prend de temps. S'il y a 500 chaînes

on arrive justement à 29 secondes. Cette procédure de garbage collection doit être utilisée constamment par l'interpréteur lors d'une opération de tri puisque le tri a pour effet de copier en permanence des chaînes de caractères. Voici maintenant à titre de comparaison les temps de tri avec la méthode bubble sort:

| n | sec. |
|---|---|
| 50 | 12.1 |
| 100 | 49 |
| 200 | 238.2 |

## 6.11 TRANSMISSION DE DONNEES A D'AUTRES ORDINATEURS

La transmission de données du CPC 464 à d'autres ordinateurs par l'intermédiaire des cassettes ou des disquettes n'est en général pas possible du fait de la différence entre les formats de stockage des divers ordinateurs. Par contre, la transmission directe de données du CPC à un autre ordinateur peut fonctionner si l'ordinateur récepteur possède un port parallèle pouvant recevoir des données comme c'est le cas par exemple de tous les ordinateurs COMMODORE.

Le CPC possède également un port parallèle qui ne peut toutefois servir qu'à l'envoi et non à la réception de données. Il s'agit du port imprimante auquel peut être connecté une imprimante Centronics. La procédure de transmission à travers un tel port est extrêmement simple et ne nécessite du côté de l'ordinateur récepteur qu'un programme assez élémentaire. Matériellement la connexion nécessite un câble reliant le port imprimante de votre CPC par exemple au port utilisateur d'un COMMODORE. Comme l'imprimante est connue du CPC, il suffit d'appeler ce port à travers le canal 8 qui est affecté par le CPC à l'imprimante. Cette méthode vous permet par exemple de transmettre les listings imprimés dans le présent ouvrage à un programme de traitement de texte tournant sur un COMMODORE. Seule la réception des données nécessite une petite routine en langage-machine telle que celle que nous vous fournissons ici pour un COMMODORE 64. Cette routine reçoit les données venant du CPC 464, les convertit en code COMMODORE et les écrit sur un fichier séquentiel sur cassette ou sur disquette, ou simplement à l'écran. Une fois que vous avez appelé cette routine sur un COMMODORE 64, vous pouvez transmettre

un programme de votre CPC au COMMODORE 64 simplement avec l'instruction:

LIST #8

Vous pouvez également transmettre tout autre texte ou caractère avec l'instruction PRINT #8:

PRINT #8, "Des nouvelles du CPC 464"

Le routine en langage-machine sur le COMMODORE 64 est conçue de façon à ce que vous puissiez mettre fin à la transmission en envoyant un code CHR$(0).

Il est toutefois important de noter que le port Centronics du CPC ne travaille qu'avec 7 bits alors qu'un port Centronics dispose normalement de 8 bits. Le bit supérieur est en effet toujours branché sur la masse. Ceci ne peut vous poser de problèmes pour transmettre des textes ASCII puisque le code ASCII est un code 7 bits. Par contre si vous voulez réaliser un hardcopy graphique de l'écran sur une imprimante, vous risquez de rencontrer certaines difficultés car la pluspart des imprimantes graphiques emploient 8 aiguilles.

Voici maintenant le programme en langage-machine pour le COMMODORE 64:

```
30:  DD00          CIA     =    $DD00
40:  DD00          PORTA   =    CIA       ; port pour 'occupé'
50:  DD01          PORTB   =    CIA+1     ; port pour données
```

```
60:   DD0D              CR       =    CIA+13   ;registre de contrôle
70:
80:   C000                       *=   $C000
90:   C000 AD 00 DD   START    LDA PORTA
100:  C003 29 FB               AND #%11111011 ;busy low
110:  C005 8D 00 DD            STA PORTA
120:  C008 A9 10               LDA #%100000
130:  C00A 2C 0D DD   WAIT     BIT CR
140:  C00D F0 FB               BEQ WAIT       ;attendre flag
150:  C00F AD 00 DD            LDA PORTA
160:  C012 09 04               ORA #%100      ;busy hi
170:  C014 8D 00 DD            STA PORTA
180:  C017 AD 01 DD            LDA PORTB      ;lire données
190:  C01A F0 2A               BEQ ENDE       ;si 0, fin
200:  C01C C9 0A               CMP #10        ;ignorer line feed
210:  C01E F0 E0               BEQ START
220:  C020 C9 41               CMP #"A"
230:  C022 90 12               BCC END        ;conversion en
240:  C024 C9 5B               CMP #"Z"+1
250:  C026 B0 04               BCS TEST       ;ASCII CBM
260:  C028 09 80               ORA #$80
270:  C02A 30 0A               BMI END
280:  C02C C9 61      TEST     CMP #$61
290:  C02E 90 06               BCC END
300:  C030 C9 7B               CMP #$7B
310:  C032 B0 02               BCS END
320:  C034 29 5F               AND #$5F
330:  C036 48         END      PHA            ;ranger les données
340:  C037 A2 01               LDX #1
350:  C039 20 C9 FF            JSR $FFC9      ;édition sur le canal 1
```

```
360: C03C 68                PLA
370: C03D 20 D2 FF          JSR $FFD2     ;sortir un caractère
380: C040 20 CC FF          JSR $FFCC     ;sortie à nouveau sur
                                          défaut
390: C043 4C 00 C0          JMP START     ;retour au début
400: C046 60         ENDE   RTS           ;fin
```

Sur le COMMODORE 64, vous devez ouvrir tout d'abord un canal de données avec le numéro logique 1:

OPEN 1,1,1,"DONNEES" pour la cassette ou

OPEN 1,8,2,"DONNEES,S,W" pour la disquette ou

OPEN 1,3

si vous voulez simplement une édition sur l'écran. Vous pouvez ensuite lancer le programme avec SYS 49152. Vous pouvez alors envoyer par exemple un listing de programme du CPC au COMMODORE avec l'instruction LIST #8. Vous pouvez mettre fin à la transmission en entrant sur votre CPC:

```
PRINT #8, CHR$(0)
```

Le COMMODORE 64 affiche alors 'READY' et vous pouvez fermer le fichier sur lequel ont été écrites les données avec l'instruction CLOSE 1. Voici enfin l'affectation des pins du câble de transmission entre les deux ordinateurs:

| Port imprimante du CPC 464 | | Port utilisateur du COMMODORE 64 |
|---|---|---|
| Pin | Signification | Pin |
| 1 | flag | B |
| 2 | D0 | C |
| 3 | D1 | D |
| 4 | D2 | E |
| 5 | D3 | F |
| 6 | D4 | H |
| 7 | D5 | J |
| 8 | D6 | K |
| 9 | D7 | L |
| 11 | BUSY | M |
| 12 | GND | N |

6.12 IMPRESSION TRANSPARENTE

Vous avez certainement remarqué que l'impression est beaucoup plus lente que l'édition sur écran. Lorsque vous utilisez un programme qui imprime certaines données au fur et à mesure que vous les entrez, l'impression risque de ralentir considérablement votre travail. La solution idéale est bien sûr l'impression transparente qui vous permet de continuer à utiliser votre ordinateur pendant que l'impression se déroule.

Nous vous avons concocté une petite routine d'impression transparente très simple à utiliser: ajoutez à votre programme les lignes de cette routine (65000 à 65240) et vous n'aurez ensuite plus qu'à affecter à la variable pr$ les données à imprimer et à appeler la routine avec GOSUB 65000 pour déclencher l'impression transparente.

Le buffer d'imprimante sera alors rempli avec pr$ et votre programme continuera sauf si le buffer est plein.

C'est pourquoi vous pouvez adapter sa dimension à vos besoins grâce à l'instruction EVERY qui contraint l'ordinateur à examiner régulièrement s'il y a quelque chose dans le buffer. Si oui, une ligne est éditée, etc...

Attention! La fin de votre programme doit absolument se présenter comme les lignes 220 et 240 de façon à ce que le contenu du buffer soit bien entièrement imprimé!

Les lignes 100 à 200 servent de démonstration. La routine que vous

pourrez intégrer dans vos programmes commence en 65000. Pour une utlisation en grandeur réelle, changez le 7 de la ligne 65200 en 8.

Voici le programme:

```
100  CLS:WINDOW #7,21,40,1,25
140  FOR i=0 TO 130
160  pr$=HEX$(i,6)
180  GOSUB 65000
200  NEXT
210  PRINT"       "
220  ON bfl GOTO 220
240  END
65000 REM fill buffer
65020 ON fl GOTO 65060
65040 DIM pr$(999):fl=1:pfc=1
65050 EVERY 30,0 GOSUB 65160
65060 IF pc=pfc THEN 65060
65080 DI:pr$(fc)=pr$:bfl=1
65100 fc=fc+1:IF fc>999 THEN fc=0
65120 pfc=fc+1:IF pfc>999 THEN pfc=0
65140 RETURN
65160 REM print buffer
65180 IF pc=fc THEN bfl=0:RETURN
65200 PRINT#7,pr$(pc)
65220 pc=pc+1:IF pc>999 THEN pc=0
65240 RETURN
```

# CHAPITRE 7 PROGRAMMES UTILISATEUR

## 7.1 INTRODUCTION

Comme il n'y a pas encore beaucoup de programmes pour le CPC 464 sur le marché et que seuls les programmes font véritablement d'un ordinateur une machine très utile, nous vous présentons quelques applications pratiques qui pourront vous rendre de grands services, à commencer par un programme de gestion de données puisque c'est là certainement une des applications les plus courantes et les plus importantes pour tout un chacun des ordinateurs.

Le CPC 464 se prête d'ailleurs admirablement bien à la gestion de données, notamment grâce à ses 80 caractères par ligne.

Les programmes que nous vous présentons peuvent tous tourner tels quels mais nous les avons abondamment commentés et nous vous indiquons pour chaque partie de ces programmes comment vous pourriez y apporter des modifications éventuelles pour les améliorer ou les adapter à vos besoins propres. L'analyse détaillée de tels programmes ainsi que leur adaptation seront pour vous une excellente manière d'apprendre à maîtriser pleinement le BASIC.

Nous vous souhaitons beaucoup de réussite: c'est en forgeant qu'on devient forgeron!

## 7.2 PROGRAMME DE TRAITEMENT DE DONNEES

Une gestion de données est un programme universel que vous pouvez adapter à différentes applications particulières, alors qu'un programme de gestion d'adresses ou de gestion de stock constitue une application spécifique.

La gestion de données est réalisée en informatique grâce à un programme de gestion de fichier qui comporte des enregistrements et des champs. Pour bien comprendre le lien entre la gestion de fichier et un traitement de données manuel classique, il suffit de comparer le fichier informatique au fichier classique, les enregistrements étant les diverses fiches en carton dont se compose le fichier classique et le champ correspondant à chaque rubrique d'une de ces fiches en carton.

Le caractère universel d'un programme de gestion de fichier se manifeste par le fait que vous pouvez déterminer librement le nombre d'enregistrements et de champs. Vous pouvez affecter à chaque champ le nom que vous souhaitez. Si vous voulez créer un fichier d'adresse avec votre programme de gestion de fichier, vous pouvez par exemple définir le premier champ 'NOM/PRENOM', le second 'RUE' et le troisième 'LOCALITE'. Vous voyez qu'un tel programme peut vraiment rendre service à tout le monde du fait de son adaptabilité. Vous pouvez même avec un peu d'imagination l'utiliser pour le traitement de texte.

Comme ce programme est relativement long nous l'avons décomposé pour l'exposé en ces différents modules. Mais bien sûr vous devrez taper directement tout le programme sur votre CPC 464.

MODULE 1 -MENU-

Le menu vous présente les différentes fonctions dont dispose le programme en associant un numéro à chacune de ces fonctions. Pour sélectionner une fonction, vous tapez le numéro correspondant puis vous appuyez sur ENTER. Chaque fois que vous avez fini d'utiliser une fonction quelconque du programme vous êtes à nouveau ramené au menu. Vous pouvez alors indiquer à l'ordinateur d'exécuter une autre tâche.

```
gestion de fichier
10 REM****************************************************************************
20 REM**************************** c o p y r i g h t ****************************
30 REM*************************** t e a m   C P C  4 6 4 *************************
40 REM*************************** d a t a   b e c k e r *************************
50 REM****************************      1 9 8 4      ****************************
60 REM****************************************************************************
70 REM Gestion de fichier version 1.0 (6.6)
80 REM****************************************************************************
90 MODE 2
100 LOCATE 20,3:PRINT"C    P    C         D    A    T    A"
110 LOCATE 20,5:PRINT STRING$(&24,"=")
120 LOCATE 12,7:PRINT"COPYRIGHT 1984 BY DATA BECKER"
130 LOCATE 15,10:PRINT"                    M E N U"
140 LOCATE 15,11:PRINT"                    -------"
150 LOCATE 15,13:PRINT"Creer un fichier          - 1 -"
160 LOCATE 15,14:PRINT"Entrer le fichier         - 2 -"
170 LOCATE 15,15:PRINT"Actualiser le fichier     - 3 -"
180 LOCATE 15,16:PRINT"Sauvegarder le fichier    - 4 -"
190 LOCATE 15,17:PRINT"Charger un fichier        - 5 -"
200 LOCATE 15,18:PRINT"Recherche                 - 6 -"
210 LOCATE 15,19:PRINT"Imprimer le fichier       - 7 -"
220 LOCATE 15,20:PRINT"Fin du Programme          - 8 -"
230 LOCATE 15,21:PRINT"Supprimer le fichier      - 9 -"
240 PRINT:PRINT:INPUT"Votre choix (1-9)       >ENTER<",a
250 ON a GOSUB 280,630,1010,1440,1570,1700,1900,2020,2070
260 GOTO 90
```

Explication du programme:

-------------------------

| | |
|---|---|
| 10-80 | Indication du copyright ainsi que du numéro de la version du programme |
| 90 | Passage en mode 80 caractères |
| 100-230 | Edition de la page menu avec l'instruction locate |
| 240 | On vous demande de choisir une des fonctions du programme |
| 250 | Saut à la routine que vous avez choisie |
| 260 | Lorsqu'un des modules du programme a été exécuté, le programme retourne à cette ligne par une instruction RETURN. Cette ligne indique à l'ordinateur de sauter au début du module menu. |

Variables employées:

--------------------

| | |
|---|---|
| a | Numéro du module du programme sélectionné |

Mode d'emploi de ce module:

---------------------------

Entrez le numéro du module sélectionné et appuyez sur ENTER.

MODULE 2 -CREATION DU FICHIER-

Ce module vous permet de définir le fichier à créer. Vous devez fournir deux indications: la dimension que doit avoir votre fichier et d'autre part quels noms devront avoir les différents champs de votre fichier. Une fois que vous avez entré ces indications, vous avez encore la possibilité de les modifier immédiatement si vous pensez avoir commis une erreur.

```
280 REM creation du fichier
290 CLS
300 PRINT STRING$(&50,"-")
310 LOCATE 15,3:PRINT"C R E A T I O N   D U   F I C H I E R"
320 PRINT STRING$(&50,"-")
330 LOCATE 1,8:INPUT"Combien d'enregistrements doit comporter le fichier";enregi
s
340 PRINT:INPUT"Combien de champs doit comporter un enregistrement";champs
350 DIM nomenr$(enregis)
360 DIM nomcha$(champs)
365 DIM contenu$(champs,enregis)
370 REM definition des champs
380 CLS
390 PRINT"Definition des champs d'un enregistrement"
400 henregis=enregis
410 hchamps=champs
420 PRINT"----------------------------------------"
430 PRINT:PRINT"(Vous avez ";champs;"champs a definir)"
440 PRINT:PRINT
450 FOR champs=1 TO hchamps
460 PRINT"Nom du champ No ";champs:INPUT"";nomcha$(champs)
470 NEXT champs
480 PRINT:INPUT"Voulez-vous apporter des modifications";modif$
490 IF LEFT$(modif$,1)="n" THEN GOTO 90
500 CLS
510 PRINT"Modification du nom d'un champ"
520 PRINT"------------------------------"
530 FOR champs=1 TO champs -1
540 PRINT"Nom";champs;":",nomcha$(champs)
550 NEXT
560 PRINT:PRINT
570 INPUT"Quel champ voulez-vous modifier";champs
580 PRINT"Ancien nom du champ "nomcha$(champs)
590 INPUT"Nouveau nom du champ ";nomcha$(champs)
600 INPUT"Voulez-vous apporter encore une modification";modif$
610 IF LEFT$(modif$,1)="n" THEN GOTO 90 ELSE GOTO 560
620 RETURN
```

Explication du programme:

-------------------------

| | |
|---|---|
| 280-320 | En-tête du module |
| 330-340 | Entrée de la taille du fichier |
| 350-360 | Les variables réservées pour les différents noms des champs du fichier sont dimensionnées en fonction de la dimension choisie du fichier |
| 400-410 | Les nombres de champs et d'enregistrements sont placés dans des variables auxiliaires (hchamps et henregis) car les variables champs et enregis seront utilisées dans des boucles où elles pourront recevoir d'autres valeurs |
| 380-440 | En-tête du sous-module DEFINITION DES CHAMPS |
| 450-470 | Boucle d'entrée des noms de champs |
| 480 | Demande si modifications nécessaires |
| 490 | Si "non", retour au menu |
| 500-520 | En-tête du sous-module MODIFICATION |
| 530-550 | Boucle d'édition des noms de champs |
| 560-620 | Routine de modification des noms avec retour au menu quand plus aucune modification n'est demandée |

Variables employées:

--------------------

| | |
|---|---|
| enregis | Nombre d'enregistrements du fichier |
| champs | Nombre de champs du fichier |
| henregis | Conserve le nombre d'enregistrements |
| hchamps | Conserve le nombre de champs |
| nomenr$(enregis) | Noms des enregistrements |
| nomcha$(champs) | Noms des champs |
| modif$ | Contient la décision oui/non |

Améliorations possibles:

------------------------

La définition du fichier sous sa forme actuelle interdit une extension future de ce fichier. Il y a donc là une modification majeure à réaliser qui remettrait en cause beaucoup de principes utilisés dans notre programme. Il est donc recommandé aux débutants qui voudraient s'attaquer à ce problème de se faire aider par des programmeurs plus confirmés. Les débutants pourraient par contre essayer d'écrire quelques lignes de programme pour contrôler les entrées et rejeter les entrées erronées ou bien essayer d'utiliser les fenêtres lors de la procédure d'entrée. On pourrait ainsi contrôler la sortie du cadre d'une zone donnée de l'écran et améliorer l'ergonomie du programme.

Mode d'emploi de ce module:

---------------------------

Le mode d'emploi est très simple puisque vous n'avez qu'à répondre aux demandes qui vous sont faites par l'ordinateur: vous entrez d'abord le nombre d'enregistrements (par exemple pour un fichier d'adresses, combien de personnes figureront dans votre fichier) puis le nombre de champs (combien de renseignements voulez-vous pouvoir porter par personne). Vous indiquez ensuite les noms des différents champs.

Vous pouvez ensuite choisir de retourner au menu (entrez n) ou d'effectuer des modifications (entrez o). Le programme affiche alors les différents champs ainsi que leurs noms respectifs et vous entrez alors le numéro du champ que vous souhaitez modifier. Vous choisissez maintenant à nouveau, soit de procéder à une autre

modification, soit de retourner au menu.

MODULE 3 -ENTRER LE FICHIER-

Ce module vous permet de remplir le fichier que vous avez défini. On vous demande d'abord de donner un nom au fichier et vous pouvez ensuite entrer toutes les données (noms, adresses, etc...) que vous voulez y voir figurer. Après chaque entrée, vous pouvez éventuellement corriger ce que vous venez d'inscrire.

```
630 CLS
640 champs=hchamps
650 enregis=henregis
660 PRINT STRING$(&50,"-")
670 LOCATE 15,3:PRINT"E N T R E R   L E   F I C H I E R"
680 PRINT STRING$(&50,"-")
690 LOCATE 1,10:PRINT"(Votre fichier a";enregis;"enregistrements et";champs;"cha
mps)"
700 PRINT:PRINT:INPUT"Quel doit etre le nom du fichier";nomfich$
710 CLS
720 REM voir ligne 365
730 FOR enregis=1 TO henregis
740 champs=hchamps
750 FOR champs=1 TO hchamps
760 WINDOW SWAP 0
770 WINDOW 1,80,1,8
780 PRINT STRING$(&50,"-")
790 PRINT"Nom du fichier :";nomfich$
800 PRINT"Maximum d'enregistrements :"henregis,"Nombre de champs :";hchamps
810 PRINT"Numero de l'enregistrement actuel :";enregis
820 PRINT"Numero du champ actuel :";champs
830 PRINT STRING$(&50,"-")
840 WINDOW 1,80,9,25
850 CLS
860 PRINT nomcha$(champs);:INPUT"";contenu$(champs,enregis)
870 NEXT champs
880 NEXT enregis
890 LOCATE 1,10
900 PRINT:INPUT"Voulez-vous apporter des modifications";modif$
910 IF LEFT$(modif$,1)="n" THEN GOTO 90
920 CLS
930 PRINT"Modification du contenu d'un champ"
940 PRINT"----------------------------------"
950 PRINT:INPUT"Quel champ voulez-vous modifier";champs
960 INPUT"Dans quel enregistrement";enregis
970 PRINT"Ancien contenu du champ "contenu$(champs,enregis)
980 INPUT"Nouveau contenu du champ ";contenu$(champs,enregis)
990 INPUT"Voulez-vous apporter encore une modification";modif$
1000 IF LEFT$(modif$,1)="n" THEN GOTO 90 ELSE GOTO 920
```

Explication du programme:

-------------------------

| | |
|---|---|
| 630-690 | En-tête du module |
| 650-660 | Les variables enregis et champs reçoivent leurs valeurs originelles |
| 700 | Entrée du nom du fichier |
| 720 | La variable contenu$ qui contiendra les données est dimensionnée |
| 730-750 | Ouverture des boucles d'entrée des données |
| 770-830 | En-tête du sous-module d'entrée qui vous informe sur les champs et enregistrements (technique des fenêtres) |
| 840-880 | Lecture des données de votre fichier |
| 890-910 | Demande si modifications nécessaires<br>Si "non", retour au menu |
| 920-1000 | Routine de modification avec retour au menu quand plus |

aucune modification n'est demandée

Variables employées:

--------------------

enregis   Nombre d'enregistrements du fichier

champs   Nombre de champs du fichier

henregis   Conserve le nombre d'enregistrements

hchamps   Conserve le nombre de champs

nomfich$   Nom du fichier

contenu$(champs,enregis)   Tableau à deux dimensions contenant les données du fichier

modif$   Contient la décision oui/non

Améliorations possibles:

------------------------

Deux améliorations principales sont possibles qui sont à la portée de tous:

Premièrement éviter que lors de l'entrée d'un enregistrement, tous les champs définis auparavant soient effacés.

Deuxièmement appeler la routine de modifications après l'entrée de chaque enregistrement.

Notez d'autre part l'emploi des fenêtres, une méthode qui rend un programme beaucoup plus agréable à utiliser et que vous pouvez étendre à toutes les parties de notre programme mais bien sûr également à tout autre programme.

Mode d'emploi de ce module:

---------------------------

Vous entrez d'abord le nom du fichier puis vous passez dans la page d'entrée ou vous voyez en permanence le nombre maximum

d'enregistrements et de champs ainsi que les numéros des champ et enregistrement actuels. Dans la partie médiane de l'écran, vous entrez les données de votre fichier champ par champ.

Vous pouvez ensuite procéder à des modifications comme dans le module précédent ou bien retourner au menu.

## MODULE 4 -ACTUALISER LE FICHIER-

Ce module vous permet de maintenir votre fichier "dans le coup". Il arrive souvent qu'on ait besoin de modifier des données d'un fichier, à la suite d'un changement d'adresse par exemple ou bien parce qu'on a rencontré certaines personnes et qu'on en a perdu d'autres de vue. C'est ce que permet donc le module 4, module d'actualisation de votre fichier.

Vous pouvez modifier les données que contient votre fichier soit en indiquant directement le champ et l'enregistrement que vous voulez modifier, soit en feuilletant votre fichier jusqu'à ce que vous rencontriez un enregistrement qu'il vous semble nécessaire de modifier.

```
010 CLS
020 PRINT STRING$(&50,"-")
030 LOCATE 15,3:PRINT"          A C T U A L I S E R   L E   F I C H I E R"
040 PRINT STRING$(&50,"-")
050 PRINT:PRINT"Modification directe du contenu d'un champ      - 1 -"
060  PRINT"Feuilleter puis modifier                       - 2 -"
070 PRINT:INPUT"Votre choix ";a
080 IF a=2 GOTO 1170
090 CLS
100 PRINT"Modification directe du contenu d'un champ"
110 PRINT:PRINT:PRINT:INPUT"Quel champ voulez-vous modifier";champs
120 INPUT"Dans quel enregistrement";enregis
130 PRINT:PRINT:PRINT"Ancien contenu du champ "contenu$(champs,enregis)
140 INPUT"Nouveau contenu du champ ";contenu$(champs,enregis)
150 INPUT"Voulez-vous apporter encore une modification";modif$
160 IF LEFT$(modif$,1)="o" THEN GOTO 1090 ELSE RETURN
170 CLS
180 champs=hchamps
190 enregis=henregis
200 FOR enregis=1 TO henregis
210 WINDOW 1,80,1,8
220 PRINT STRING$(&50,"-")
230 PRINT"Fichier actuel :"nomfich$
240 PRINT"Enregistrement actuel :"enregis
250 PRINT STRING$(&50,"-")
260 GOSUB 1290
270 PRINT:INPUT"Voulez-vous apporter des modifications";modif$
```

```
1275 IF LEFT$(modif$,1)="n" THEN GOTO 1420
1280 GOTO 1350
1290 WINDOW 1,80,9,20
1300 CLS
1310 FOR champs=1 TO hchamps
1320 PRINT nomcha$(champs);":";contenu$(champs,enregis)
1330 NEXT champs
1340 RETURN
1350 WINDOW 1,80,21,25
1360 CLS
1370 PRINT STRING$(&50,"-")
1380 PRINT:INPUT"Quel champ voulez-vous modifier";champs
1390 INPUT"Nouveau contenu du champ ";contenu$(champs,enregis)
1400 INPUT"Voulez-vous apporter encore une modification";modif$  $
1410 IF LEFT$(modif$,1)="o" THEN GOTO 1360
1420 NEXT enregis
1430 RETURN
```

Explication du programme:

-------------------------

1010-1060 En-tête du module

1070 Choix du sous-module

1080 Saut au sous-module choisi

1090-1140 Modification directe après indication du champ et de l'enregistrement à modifier

1150-1160 Vous indiquez si vous voulez apporter encore une modification ou revenir au menu

1170-1250 En-tête de l'autre sous-module

1200 Boucle d'affichage des enregistrements

1260 Saut à la routine d'affichage des champs

1290-1330 Routine qui affiche par une boucle le nom et le contenu de chaque champ d'un enregistrement

1340 Retour en 1270

1270 Décision si modification ou saut en 1350

1350-1400 Routine de modification

1400-1410 Décision si un autre champ doit être modifié. Si "non", enregistrement suivant

Variables employées:

---

| | |
|---|---|
| a | Numéro du sous-module à appeler |
| enregis | Nombre d'enregistrements du fichier |
| champs | Nombre de champs du fichier |
| henregis | Conserve le nombre d'enregistrements |
| hchamps | Conserve le nombre de champs |
| nomfich$ | Nom du fichier |
| contenu$(champs,enregis) | Tableau à deux dimensions contenant les données du fichier |
| modif$ | Contient la décision oui/non |

Améliorations possibles:

---

C'est uniquement le deuxième module qu'il serait nécessaire d'améliorer: vous ne pouvez en effet retourner au menu que lorsque vous avez feuilleté l'ensemble des enregistrements, ce qui peut être extrêmement pénible si le fichier est de taille importante. Il faudrait donc écrire une petite routine qui teste en permanence si une touche (par exemple '&') est enfoncée et dans ce cas vous renvoie au menu avec une simple instruction GOTO.

Mode d'emploi de ce module:

---

Vous devez d'abord décider si vous voulez faire une modification

directe ou feuilleter d'abord les divers enregistrements de votre fichier. Vous ne pouvez évidemment faire de modification directe que si vous connaissez les numéros du champ et de l'enregistrement à modifier. Dans le cas contraire, vous devez choisir le deuxième sous-module.

Si vous choisissez la modification directe, vous entrez les numéros du champ et de l'enregistrement à modifier. Le programme affiche alors l'ancien contenu de ce champ que vous pouvez maintenant modifier. Vous indiquez ensuite si vous voulez procéder à une autre modification ou revenir au menu.

Si vous choisissez de feuilleter le contenu du fichier, le programme affiche le contenu des divers champs de chaque enregistrement l'un après l'autre. Vous pouvez alors modifier un champ en en indiquant le numéro. Vous pouvez ensuite modifier un autre champ ou consulter l'enregistrement suivant.

MODULE 5 -SAUVEGARDER LE FICHIER-

Ce module vous permet de sauvegarder votre fichier sur cassette de façon à ce que vos données soient conservées une fois que vous aurez éteint votre ordinateur.

```
1440 CLS
1450 PRINT STRING$(&50,"-")
1460 PRINT"                    S A U V E G A R D E R    L E    F I C H I E R"
1470 PRINT STRING$(&50,"-")
1480 LOCATE 5,10:PRINT"Je sauvegarde votre fichier, veuillez patienter."
1490 OPENOUT nomfich$
1500 FOR enregis=1 TO henregis
1510 FOR champs=1 TO hchamps
1520 PRINT£9,contenu$(champs,enregis)
1530 NEXT champs
1540 NEXT enregis
1550 CLOSEOUT
1560 RETURN
```

Explication du programme:

-------------------------

1440-1470 En-tête du module

1480 Message à l'utilisateur

1490 Ouverture d'un fichier sur cassette avec le nom du fichier se trouvant en mémoire

1500-1540 Boucle d'écriture des données sur la bande

1550 Fermeture du fichier

1560 Retour au menu

Variables employées:

--------------------

| | |
|---|---|
| enregis | Nombre d'enregistrements du fichier |
| champs | Nombre de champs du fichier |
| henregis | Conserve le nombre d'enregistrements |
| hchamps | Conserve le nombre de champs |
| nomfich$ | Nom du fichier |
| contenu$(champs,enregis) | Tableau à deux dimensions contenant les données du fichier |

Améliorations possibles:

------------------------

Il n'y a pas ici d'améliorations essentielles à apporter. Vous pouvez toutefois améliorer la présentation et l'ergonomie de ce module.

Mode d'emploi de ce module:

---------------------------

Le message vous invitant à appuyer sur les touches PLAY et RECORD du lecteur de cassette apparaît peu après que vous ayez sélectionné ce module. Vous n'avez plus ensuite qu'à appuyer sur une touche quelconque pour que la sauvegarde s'effectue.

MODULE 6 -CHARGER UN FICHIER-

Ce module vous permet de charger les fichiers que vous avez sauvegardés sur cassette. Mais vous devez auparavant fixer les paramètres de nombres de champs et d'enregistrements en fonction du fichier que vous allez charger. Utilisez pour celà le module 1.

```
1570 CLS
1580 PRINT STRING$(&50,"-")
1590 PRINT"                    C H A R G E R         U N       F I C H I E R" 2"
1600 PRINT STRING$(&50,"-")
1610 LOCATE 5,10:PRINT"Je charge votre fichier, veuillez patienter."
1620 OPENIN "
1630 FOR enregis=1 TO henregis
1640 FOR champs=1 TO hchamps
1650 INPUT£9,contenu$(champs,enregis)
1660 NEXT champs
1670 NEXT enregis
1680 CLOSEIN
1690 RETURN
```

Explication du programme:

-------------------------

1570-1600 En-tête du module

1610 Message à l'utilisateur

1620 Ouverture d'un fichier sur cassette en lecture

1630-1670 Boucle de lecture des données sur la bande

1680 Fermeture du fichier

1690 Retour au menu

Variables employées:

--------------------

| | |
|---|---|
| enregis | Nombre d'enregistrements du fichier |
| champs | Nombre de champs du fichier |
| henregis | Conserve le nombre d'enregistrements |
| hchamps | Conserve le nombre de champs |
| contenu$(champs,enregis) | Tableau à deux dimensions contenant les données du fichier |

Améliorations possibles:

------------------------

Ici aussi, vous pouvez toutefois améliorer la présentation et l'ergonomie de ce module. Il serait d'autre part très intéressant que le programme puisse charger un fichier sans en connaître d'avance les nombres de champs et d'enregistrements.

Mode d'emploi de ce module:

---------------------------

Le message vous invitant à appuyer sur la touche PLAY du lecteur de cassette apparaît peu après que vous ayez sélectionné ce module. Enroulez la cassette jusqu'au début du fichier à charger puis appuyez sur la touche PLAY. Vous n'avez plus ensuite qu'à appuyer sur une touche quelconque pour que le chargement s'effectue.

MODULE 7 -RECHERCHER-

Ce module vous permet de rechercher une donnée figurant dans votre fichier, un nom ou un numéro de téléphone par exemple. Il vous suffit d'indiquer quelle est la donnée que vous voulez retrouver et combien de lettres de cette donnée doivent être prises en compte. Plus le nombre de lettres significatives est réduit, plus la recherche sera rapide mais plus il y aura de chances également que la donnée trouvée ne corresponde pas à ce que vous cherchez.

```
1700 CLS
1710 PRINT STRING$(&50,"-")
1720 PRINT"                              R E C H E R C H E R"
1730 PRINT STRING$(&50,"-")
1740 LOCATE 5,10:INPUT"Quel est le critere de recherche ";critere$
1750 LOCATE 5,11:INPUT"Combien de lettres caracteristiques ";lettres
1760 PRINT:PRINT:PRINT"La recherche est en cours, veuillez patienter"
1770    PRINT"=========================================="
1780 cherche$=LEFT$(critere$,lettres)
1790 FOR enregis=1 TO henregis
1800 FOR champs=1 TO hchamps
1810 aide$=contenu$(champs,enregis)
1820 trouve$=LEFT$(aide$,lettres)
1830 IF cherche$=trouve$ THEN GOTO 1880
1840 NEXT champs
1850 NEXT enregis
1860 INPUT"Ce critere n'a pas ete trouve, faut-il en chercher un autre";choix$
1870 IF LEFT$(choix$,1)="n" THEN RETURN ELSE GOTO 1700
1880 PRINT"Le critere de recherche ";critere$;" se trouve dans le champ ";champs
;" de l'enregistrement ";enregis:INPUT"Rechercher en
core une fois ";choix$
1890 IF LEFT$(choix$,1)="n" THEN RETURN ELSE GOTO 1700
```

Explication du programme:

-------------------------

1700-1730 En-tête du module

1740-1750 Entrée des informations concernant le critère de recherche

1760-1770 Informations pour l'utilisateur

1780 Les lettres significatives du critère de recherche sont affectées à la variable cherche$

1790-1800 Ouverture de la boucle de recherche

1810 Les divers contenus du fichier sont affectés à la variable aide$

1820 Les lettres significatives des contenus du fichier sont affectées à la variable trouve$

1830 Si cherche$=trouve$, la donnée recherchée a été trouvée

1840-1850 Fermeture des deux boucles

1860-1870 Si le programme n'est pas sorti des boucles, c'est que la donnée recherchée n'a pas été trouvée. Un message l'indique. Vous pouvez ensuite faire rechercher une autre donnée ou retourner au menu

1880-1890 C'est ici que le programme saute si la donnée recherchée a été trouvée. Le critère de recherche ainsi que les numéros des champ et enregistrement où il a été trouvé sont affichés. Vous pouvez alors faire faire une autre recherche ou retourner au menu

Variables employées:

--------------------

critere$ Critère de recherche

lettres Nombre de lettres significatives

cherche$ Lettres significatives du critère de recherche

| | |
|---|---|
| aide$ | Variable auxiliaire à laquelle sont affectés les uns après les autres les divers contenus du fichier |
| trouve$ | Lettres significatives des données du fichier à comparer avec cherche$ |
| enregis | Nombre d'enregistrements du fichier |
| champs | Nombre de champs du fichier |
| henregis | Conserve le nombre d'enregistrements |
| hchamps | Conserve le nombre de champs |
| choix$ | Décision oui/non |

Améliorations possibles:
------------------------

Il n'est pas nécessaire d'améliorer la routine de recherche qui est parfaitement adaptée au CPC 464 et travaille très vite, pour une routine BASIC, même avec des masses importantes de données.
Vous pourriez par contre faire afficher non pas simplement les numéros de champ et d'enregistrement de la donnée recherchée mais l'enregistrement complet avec le contenu de tous ses champs. Ceci est à la portée de tous les programmeurs, y compris les débutants.

Mode d'emploi de ce module:
---------------------------

Vous devez entrer d'abord le critère de recherche puis le nombre de lettres significatives. Lorsque le programme a trouvé le critère de recherche, il affiche les numéros de champ et d'enregistrement correspondants. Vous pouvez ensuite, comme pour les autres modules, indiquer si vous souhaitez effectuer une nouvelle recherche (touche 'o') ou retourner au menu.
Si le critère de recherche n'a pas été trouvé, le programme vous l'indique et vous pouvez également indiquer si vous souhaitez

effectuer une nouvelle recherche (touche 'o') ou retourner au menu. Il peut être intéressant, si votre critère de recherche n'a pas été retrouvé d'effectuer une nouvelle recherche avec un nombre inférieur de lettres significatives. Ceci peut en effet vous permettre de retrouver la donnée que vous cherchez même si vous aviez fait une légère faute de frappe lorsque vous l'aviez entrée. Il existe d'autre part dans le commerce des programmes comprenant des routines de recherche avec une certaine "tolérance d'erreurs" mais ces programmes sont en général très onéreux.

MODULE 8 -IMPRIMER LE FICHIER-

Ce module vous permet de faire imprimer votre fichier.

```
1900 CLS
1910 PRINT STRING$(&50,"-")
1920 PRINT"                    I M P R I M E R      L E     F I C H I E R"
1930 PRINT STRING$(&50,"-")
1940 LOCATE 5,10:PRINT"Veuillez patienter pendant l'impression du fichier."
1950 FOR enregis=1 TO henregis
1960 FOR champs=1 TO hchamps
1970 PRINT£8,contenu$(champs,enregis)
1980 NEXT champs
1990 NEXT enregis
2000 PRINT:INPUT"Voulez-vous une autre impression";choix$
2010 IF LEFT$(choix$,1)="n" THEN RETURN ELSE GOTO 1950
```

Explication du programme:

-------------------------

1900-1930 En-tête du module

1940 Message à l'utilisateur

1950-1990 Boucle d'impression des données

2000-2010 Possibilité d'imprimer à nouveau ou retour au menu

Variables employées:

--------------------

enregis Nombre d'enregistrements du fichier

champs Nombre de champs du fichier

henregis Conserve le nombre d'enregistrements

hchamps Conserve le nombre de champs

Améliorations possibles:

-------------------------

La routine actuelle est relativement simple: les données du fichier sont imprimées les unes après les autres sans aucun format.

Une impression formatée afficherait par contre les numéros de champ et d'enregistrement ainsi que les noms des champs. Une telle impression devrait également prévoir des espaces entre les différents enregistrements. Ces modifications sont assez simples à réaliser grâce à l'excellente instruction PRINT USING de votre CPC 464. Nous vous conseillons de lire attentivement la section de votre manuel qui est consacrée à cette instruction, ceci vous épargnera beaucoup de travail de manipulation avec les fonctions alphanumériques (right$, left$ ...).

Mode d'emploi de ce module:

---------------------------

Le mode d'emploi de ce module ne présente aucune difficulté, il vous suffit de répondre aux messages de votre programme.

MODULE 9 -FIN DU PROGRAMME-

Ce module vous permet de mettre fin au programme.

```
2020 CLS
2030 PRINT STRING$(&50,"-")
2040 PRINT"                F I N        D U       P R O G R A M M E"
2050 PRINT STRING$(&50,"-")
2060 LOCATE 5,10:INPUT"Etes-vous certain de vouloir mettre fin au Programme";b$:
IF LEFT$(b$,1)="o" THEN END ELSE RETURN
```

Explication du programme:

-------------------------

2020-2050 En-tête du module

2060 On vous demande de confirmer que vous voulez bien mettre fin au programme. Si "non", vous retournez au menu

Variables employées:

--------------------

b$ Décision oui/non

Améliorations possibles:

------------------------

Les seules améliorations possibles concernent la présentation.

Mode d'emploi de ce module:

---------------------------

Le mode d'emploi de ce module ne présente pas non plus de difficulté, il vous suffit soit de confirmer que vous voulez mettre fin au programme, soit d'indiquer que vous souhaitez revenir au menu.

MODULE 10 -SUPPRIMER LE FICHIER-

Ce module vous permet de supprimer des données. Ceci est très utile pour vous permettre d'actualiser votre fichier en remplaçant des enregistrements dont vous n'avez plus besoin par de nouvelles données. Il faut bien sûr toujours être prudent quand on utilise ce genre de routine car il est toujours plus rapide de supprimer une donnée que de la réinscrire. A cet égard, il est bien sûr très intéressant de faire imprimer votre fichier avant de supprimer des informations ce qui vous facilitera le travail si vous supprimez par erreur des informations dont finalement vous avez toujours besoin.

Explication du programme:

-------------------------

2070-2120 En-tête du module

2130-2140 Vous choisissez un des sous-modules

2150-2200 Routine de suppression de toutes les données avec l'instruction CLEAR après vous avoir demandé confirmation

2220-2230 En-tête du sous-module

2240 Entrée du numéro de l'enregistrement à supprimer

2250 Ouverture d'une boucle de suppression des champs

2260 Suppression des données par affectation de chaînes vides aux variables de contenu

2270 Fin de la boucle

2280-2290 Vous choisissez entre supprimer un autre enregistrement et retourner au menu

Variables employées:

--------------------

enregis  Nombre d'enregistrements du fichier

champs  Nombre de champs du fichier

henregis  Conserve le nombre d'enregistrements

hchamps  Conserve le nombre de champs

contenu$(champs,enregis)  Tableau à deux dimensions contenant les données du fichier

choix$  Décision oui/non

Améliorations possibles:

------------------------

Il n'y a pas d'améliorations essentielles à apporter en ce qui concerne la suppression complète du fichier. Vous pouvez toutefois améliorer la présentation et l'ergonomie de ce module. Par contre, en ce qui concerne la suppression sélective, il serait intéressant de pouvoir ne supprimer qu'un seul champ d'un enregistrement. Vous pourrez certainement réaliser assez aisément cette modification en permettant à l'utilisateur d'indiquer le numéro du champ à supprimer s'il ne veut pas supprimer la totalité de l'enregistrement.

Mode d'emploi de ce module:

---------------------------

Vous devez tout d'abord indiquer si vous voulez une suppression complète ou sélective de données.

Si vous choisissez la suppression totale, le programme vous demande confirmation. Si vous entrez un "o" les données sont supprimées, sinon vous êtes ramené au menu.

Si vous choisissez la suppression sélective, vous devez entrer le numéro de l'enregistrement à supprimer. L'enregistrement une fois supprimé, vous pouvez choisir entre une nouvelle suppression et le retour au menu.

```
2070 CLS
2080 PRINT STRING$(&50,"-")
2090 PRINT"                    A N N U L A T I O N"
2100 PRINT STRING$(&50,"-")
2110 LOCATE 10,10:PRINT"Annulation totale                - 1 -"
2120 LOCATE 10,11:PRINT"Annulation d'un enregistrement - 2 -"
2130 LOCATE 10,14:INPUT"Votre choix :";choix$
2140 IF choix$="2" GOTO 2210
2150 CLS
2160 PRINT"Annulation totale"
2170 PRINT"-----------------"
2180 LOCATE 5,10:INPUT"Etes-vous certain de vouloir tout supprimer";choix$
2190 IF LEFT$(choix$,1)="o" THEN CLEAR
2200 GOTO 90
2210 CLS
2220 PRINT"Annulation d'un enregistrement"
2230 PRINT"------------------------------"
2240 LOCATE 5,10:INPUT"Quel enregistrement voulez-vous supprimer ";enregis
2250 FOR champs=1 TO hchamps
2260 contenu$(champs,enregis)="      "
2270 NEXT champs
2280 LOCATE 5,15:INPUT"Voulez-vous supprimer un autre enregistrement ";choix$
2290 IF LEFT$(choix$,1)="o" THEN GOTO 2210 ELSE RETURN
```

## 7.3 TRAITEMENT DE TEXTE

Nous vous fournissons dans ce chapitre un second programme constituant une application de type professionnel grâce auquel votre CPC 464 pourra être plus qu'un simple moyen de vous initier à l'informatique, un véritable outil de travail.

Il s'agit d'une autre application classique depuis de nombreuses années des ordinateurs, le traitement de texte. Ce programme vous permettra notamment de sauvegarder sur cassette les textes que vous avez tapés, de façon à ce que vous puissiez plus tard les consulter, les modifier ou les compléter et bien sûr également les imprimer. L'intérêt du traitement de texte par rapport à la machine à écrire est bien sûr également de vous permettre de modifier à l'écran toute faute de frappe, sans laisser aucune trace lors de l'impression.

Nous ne nous étendrons pas plus longuement sur les avantages considérables que présente le traitement de texte mais nous sommes persuadés que vous les découvrirez très vite dès que vous serez familiarisé avec ce programme.

Si notre programme ne peut pas d'une manière générale rivaliser avec les traitements de texte commerciaux existant sur d'autres ordinateurs, nous pensons qu'il est néammoins très utile et que notamment le module d'entrée du texte est assez réussi puisque vous pouvez entrer votre texte librement en n'importe quel endroit de l'écran ce qui n'est normalement pas le cas pour des programmes BASIC de cette taille.

Comme pour la gestion de fichier nous vous fournissons des explications et commentaires détaillés pour chaque partie du programme de façon à ce que vous puissiez d'une part utiliser aisément notre programme et d'autre part y apporter les modifications que vous souhaitez.

MODULE 1 -MENU-

Le menu vous présente les différentes fonctions dont dispose le programme en associant un numéro à chacune de ces fonctions. Pour sélectionner une fonction, vous tapez le numéro correspondant puis vous appuyez sur ENTER. Chaque fois que vous avez fini d'utiliser une fonction quelconque du programme vous êtes à nouveau ramené au menu. Vous pouvez alors indiquer à l'ordinateur d'exécuter une autre tâche.

```
10 REM*****************************************************************************
20 REM***************************** c o p y r i g h t ****************************
30 REM*************************** t e a m   C P C  4 6 4 *************************
40 REM*************************** d a t a   b e c k e r  *************************
50 REM*****************************       1 9 8 4        ****************************
60 REM*****************************************************************************
70 REM Traitement de texte version 1.0 (6.6)
80 REM*****************************************************************************
90 DIM contenu$(82,25)
100 MODE 2
110 LOCATE 20,3:PRINT"C    P    C          T    E    X    T"
120 LOCATE 20,5:PRINT STRING$(&24,"=")
130 LOCATE 12,7:PRINT"COPYRIGHT 1984 BY DATA BECKER"
140 LOCATE 15,10:PRINT"                   M E N U"
150 LOCATE 15,11:PRINT"                   -------"
160 LOCATE 15,13:PRINT"Creer un texte              - 1 -"
170 LOCATE 15,14:PRINT"Examiner le texte           - 2 -"
180 LOCATE 15,15:PRINT"Modifier le texte           - 3 -"
190 LOCATE 15,16:PRINT"Supprimer le texte          - 4 -"
200 LOCATE 15,17:PRINT"Sauvegarder un texte        - 5 -"
210 LOCATE 15,18:PRINT"Charger un texte            - 5 -"
220 LOCATE 15,19:PRINT"Imprimer le texte           - 7 -"
230 LOCATE 15,20:PRINT"Fin du Programme            - 8 -"
240 PRINT:PRINT:INPUT"Votre choix (1-8)      >ENTER<";a
250 ON a GOSUB 260,680,850,1010,1080,1240,1390,1540
```

Explication du programme:

-------------------------

| | |
|---|---|
| 10-80 | Indication du copyright ainsi que du numéro de la version du programme |
| 90 | Dimensionnement des variables devant contenir le texte |
| 100 | Passage en mode 80 caractères |
| 110-230 | Edition de la page menu avec l'instruction locate |
| 240 | On vous demande de choisir une des fonctions du programme |
| 250 | Saut à la routine que vous avez choisie |

Variables employées:

--------------------

| | |
|---|---|
| a | Numéro du module du programme sélectionné |

Mode d'emploi de ce module:

---------------------------

Entrez le numéro du module sélectionné et appuyez sur ENTER.

MODULE 2 -CREATION DU TEXTE-

Ce module vous permet d'entrer votre texte.

Vous pouvez entrer votre texte librement en utilisant le clavier de votre ordinateur comme celui d'une machine à écrire. Le curseur (un petit point) vous indique où s'affichera le prochain caractère. Vous pouvez déplacer ce curseur avec les touches de contrôle du curseur.

```
260 CLS
270 PRINT STRING$(&50,"-")
280 LOCATE 15,3:PRINT"C R E A T I O N   D U   T E X T E"
290 PRINT STRING$(&50,"-")
300 LOCATE 10,10:INPUT"Comment doit s'appeler le texte ";nomtexte$:a$
310 CLS
320 ligne=1:colonne=1
330 PLOT colonne*8+8,400-ligne*16+8,1
340 a$=inkey$:if a$="" goto 330
350 plot colonne*8+8,400-ligne*16+8,1
360 if a$=chr$(13) goto 470
370 if a$=chr$(93) goto 630
380 if a$=chr$(240) goto 510
390 if a$=chr$(241) goto 540
400 IF a$=CHR$(242) GOTO 570
410 IF a$=CHR$(243) GOTO 600
420 colonne=colonne+1
430 IF colonne=80 THEN GOTO 650
440 contenu$(colonne,ligne)=a$
450 LOCATE colonne,ligne:PRINT a$
460 GOTO 330
470 colonne=1
480 ligne=ligne+1
490 IF ligne>24 THEN ligne=24
500 GOTO 330
510 ligne=ligne-1
520 IF ligne<1 THEN ligne=1
530 GOTO 330
540 ligne=ligne+1
550 IF ligne>24 THEN ligne=24
560 GOTO 330
```

```
570 IF colonne=1 THEN ligne=ligne-1:colonne=80
580 colonne=colonne-1
590 GOTO 330
600 IF colonne=80 THEN ligne=ligne+1:colonne=1
610 colonne=colonne+1
620 GOTO 330
630 contenu$(colonne,ligne)=CHR$(93)
640 GOTO 100
650 ligne=ligne+1:colonne=1
660 IF ligne>24 THEN ligne=24
670 GOTO 330
```

Explication du programme:

-------------------------

| | |
|---|---|
| 260 | Vidage de l'écran |
| 270-290 | Edition de l'en-tête du module |
| 300 | Entrée du nom du texte |
| 310 | Vidage de l'écran |
| 320 | Les variables de position ligne et colonne sont initialisées à 1. Le curseur se trouve ainsi placé dans l'angle supérieur gauche de l'écran. Ces variables indiquent ensuite en permanence la position du curseur sur l'écran |
| 330 | Le curseur est placé dans l'emplacement suivant de l'écran |
| 340 | Lecture d'un caractère entré au clavier |
| 360-410 | Interrogation respectivement des touches suivantes: ENTER, crochet fermé (]) et touches de contrôle du curseur. Si vous appuyez sur l'une de ces 6 touches, le programme saute à la routine correspondante |
| 420 | La valeur de la variable colonne est augmentée de 1 pour |

qu'elle corresponde à la nouvelle position du curseur

430 Saut à une routine spéciale si la dernière colonne (80) est atteinte

440 Le caractère qui a été entré et placé dans a$ est transmis à la variable contenu$(colonne,ligne) correspondant à l'emplacement du curseur

450 Affichage du caractère dans l'emplacement de l'écran qui convient

460 Retour pour lire le prochain caractère

470-500 Routine de traitement de la touche ENTER. Si vous appuyez sur cette touche, le curseur saute au début de la ligne suivante. La ligne 490 contrôle qu'on ne sort pas de l'écran

510-530 Routine de traitement de la touche curseur haut. Si vous appuyez sur cette touche, la valeur de la variable ligne est diminuée de 1. La ligne 520 contrôle qu'on ne sort pas de l'écran

540-560 Routine de traitement de la touche curseur bas. Si vous appuyez sur cette touche, la valeur de la variable ligne est augmentée de 1. La ligne 550 contrôle qu'on ne sort pas de l'écran

570-590 Routine de traitement de la touche curseur gauche. Si vous appuyez sur cette touche, le programme contrôle tout d'abord si le curseur ne se trouvait pas en début de ligne. Dans ce cas, la valeur de la variable ligne est diminuée de 1 et la variable colonne passe à 80. Si ce n'est pas le cas, la variable colonne est simplement diminuée de 1

600-620 Routine de traitement de la touche curseur droite. Si vous appuyez sur cette touche, le programme contrôle

tout d'abord si le curseur ne se trouvait pas en fin de ligne. Dans ce cas, la valeur de la variable ligne est augmentée de 1 et la variable colonne passe à 1. Si ce n'est pas le cas, la variable colonne est simplement augmentée de 1

630-640 Routine de traitement de la touche crochet fermé (]). Un caractère est tout d'abord ajouté dans le dernier emplacement du curseur de façon à ce que la fin du texte puisse être plus tard localisée. Puis le programme saute au menu

650-670 Routine de passage à la ligne lorsque vous arrivez en écrivant en fin de ligne. La ligne 660 contrôle qu'on ne sort pas de l'écran

Variables employées:

--------------------

| | |
|---|---|
| nomfich$ | Nom du texte |
| ligne | Numéro de ligne actuel du curseur |
| colonne | Numéro de colonne actuel du curseur |
| a$ | Contient le dernier caractère entré |
| contenu$(colonne,ligne) | Contient tout le texte, chaque lettre étant définie par ses numéros de colonne et de ligne |

Améliorations possibles:

------------------------

La routine d'entrée n'a pas en fait à être améliorée notablement. Tout au plus pourrait-on essayer d'utiliser plus de touches de contrôle, les touches DEL et CLR par exemple. On pourrait par exemple avoir une touche qui efface tout le texte à l'écran.

Il serait cependant possible d'apporter une amélioration substantielle en ce qui concerne la longueur possible du texte. Pour le moment le texte ne peut comporter que 24 lignes. Il serait donc possible de donner une dimension supérieure à la variable contenu$ ainsi qu'à tous les paramètres actuellement fixés à 24. Il faut cependant faire attention à ne pas surcharger la mémoire de l'ordinateur. Il faudrait également, et c'est déjà plus difficile, modifier en conséquence les routines de modification et d'examen du texte.

Mode d'emploi de ce module:
---------------------------

Vous devez tout d'abord entrer le nom du texte (lettre à X, lettre d'affaire du 2.3.1998, etc...). Vous vous trouvez ensuite dans la page d'entrée. Vous pouvez alors entrer votre texte tout à fait normalement et utiliser les fonctions suivantes: ENTER pour passer au début de la ligne suivante, les touches curseur pour déplacer le curseur sur toute la surface de l'écran et enfin crochet fermé pour sortir du module d'entrée du texte et retourner au menu. Nous vous invitons cependant instamment à ne pas utiliser la touche crochet ouvert que nous avons utilisée comme marque automatique de la fin du texte entré.

Les touches DEL et CLR n'ont pas de fonction dans la version actuelle du programme mais elles peuvent produire des caractères indéfinis si vous les utilisez en liaison avec la touche SHIFT. Dans ce cas, il vous suffit de ramener le curseur sur ces caractères indésirables et de les effacer avec la touche espace ou avec tout autre caractère.

## MODULE 3 -EXAMINER LE TEXTE-

Ce module vous permet de consulter à l'écran le texte se trouvant en mémoire. Ceci vous permet notamment de consulter un texte que vous venez juste de charger en mémoire et dont vous ne vous rappelez pas exactement mais aussi par exemple, si vous placez votre moniteur dans la vitrine de votre magasin, d'afficher un texte publicitaire. Songez que dans ce dernier cas il vous sera très aisé de modifier régulièrement le texte ainsi exposé.

```
680 CLS
690 PRINT STRING$(&50,"-")
700 LOCATE 15,3:PRINT"          E X A M I N E R   L E   T E X T E"
710 PRINT STRING$(&50,"-")
720 LOCATE 10,10:PRINT"Frappez une touche pour examiner le texte"
730 CALL &BB18
740 CLS
750 FOR ligne=1 TO 24
760 FOR colonne=1 TO 80
770 a$=contenu$(colonne,ligne)
780 IF a$="" THEN a$=CHR$(32)
790 IF a$=CHR$(93) THEN GOTO 830
795 contenu$(colonne,ligne)=a$
800 LOCATE colonne,ligne:PRINT a$
810 NEXT colonne
820 NEXT ligne
830 CALL &BB18
    840 GOTO 100
```

Explication du programme:

--------------------------

680 Vidage de l'écran

690-710 En-tête du module

720 Message à l'utilisateur

730 Routine qui arrête le programme jusqu'à ce que vous appuyiez sur une touche

740 Vidage de l'écran

750-760 Ouverture de la boucle permettant d'appeler tous les emplacements de l'écran

770 La variable a$ reçoit tous les caractères entrés

780 Interroge si l'emplacement de l'écran est vide. Si oui, un caractère espace y est affiché

790 Interroge s'il s'agit du caractère EOF (fin du texte). Si oui, fin de l'édition

795 Intégration du caractère espace dans le texte

800 Affichage du caractère actuel dans l'emplacement qui convient

810-820 Fermeture de la boucle

830 L'écran est figé tant que vous n'appuyez sur aucune touche. Dès que vous appuyez sur une touche, retour au menu

840 Retour au menu

Variables employées:

--------------------

| | |
|---|---|
| ligne | Comme au module précédent |
| colonne | Comme au module précédent |
| a$ | Comme au module précédent |
| contenu$ | Comme au module précédent |

Mode d'emploi de ce module:

---------------------------

Vous devez appuyer sur une touche comme un message vous y invite. Vous voyez alors le texte apparaître à l'écran et vous n'avez plus qu'à appuyer à nouveau sur une touche dès que vous souhaitez revenir au menu.

Nous vous conseillons vivement de toujours appeler cette routine lorsque vous venez d'entrer un texte et avant de faire appel à toute autre fonction.

MODULE 4 -MODIFIER LE TEXTE-

Ce module vous permet de modifier votre texte. Nous vous invitons à lire dans l'explication du programme quelle technique nous avons employée pour économiser le plus de place en mémoire possible.

```
850 CLS
860 PRINT STRING$(&50,"-")
870 LOCATE 15,3:PRINT"          M O D I F I E R   L E   T E X T E"
880 PRINT STRING$(&50,"-")
890 LOCATE 10,10:PRINT"Frappez une touche pour modifier le texte"
900 CALL &BB18
910 CLS
920 FOR ligne=1 TO 24
930 FOR colonne=1 TO 80
940 a$=contenu$(colonne,ligne)
950 IF a$="" THEN a$=CHR$(32)
960 IF a$=CHR$(91) THEN GOTO 1000
970 LOCATE colonne,ligne:PRINT a$
980 NEXT colonne
990 NEXT ligne
1000 GOTO 320
```

Explication du programme:

-------------------------

| | |
|---|---|
| 850 | Vidage de l'écran |
| 860-880 | En-tête du module |
| 890 | Message à l'utilisateur |
| 900 | Le programme attend que vous appuyiez sur une touche |
| 910 | Vidage de l'écran |
| 920-990 | Routine d'édition du texte à l'écran |
| 1000 | Saut à la routine d'entrée du module "CREATION DU TEXTE". En effet la modification du texte ne nécessite aucune fonction qui ne nous soit pas déjà offerte par le module d'entrée du texte. Il n'y a donc aucune raison de ne pas réutiliser cette routine. |

Nous vous conseillons de toujours concevoir votre programme de façon à ce que le plus grand nombre de routines possible accomplissent des tâches suffisamment générales pour pouvoir être appelées par diverses parties de vos programmes. En effet ceci vous permet de gagner de la place en mémoire mais également de gagner beaucoup de temps lors du développement de votre programme. Car d'une part vous avez alors moins de lignes à taper et d'autre part, moins vous avez de routines différentes dans vos programmes, moins vous perdrez de temps lors du dépistage des erreurs qui constitue certainement la partie la plus longue et la moins agréable du travail de programmation. Cette technique est utilisée de manière systématique par les programmmeurs professionnels pour qui le temps de programmation représente bien sûr de l'argent.

Variables employées:

--------------------

Ce sont les mêmes variables que pour les modules précédents.

Améliorations possibles:

------------------------

Les améliorations possibles ne seraient pas nécessairement simples à mettre en oeuvre et nous vous conseillons de ne vous y attaquer que si vous êtes sûr d'avoir bien compris la structure et le fonctionnement de l'ensemble du programme. Il s'agirait essentiellement de permettre des insertions de texte ou des suppressions par une seule commande de mots ou de phrases entiers. Le fait que le texte soit stocké dans une seule variable ne rend pas celà trop difficile.

Mode d'emploi de ce module:

---------------------------

Vous devez d'abord appuyer sur une touche puis vous voyez s'afficher le texte à l'écran. Pour le reste vous pouvez vous reporter au mode d'emploi du module "CREATION DU TEXTE". Vous n'avez donc plus qu'à amener le curseur sur les endroits où vous avez fait des fautes de frappe et à réécrire les caractères corrects par deçsus les lettres erronnées.

MODULE 5 -SUPPRIMER LE TEXTE-

Ce module vous permet de supprimer votre texte si vous vous apercevez après l'avoir consulté grâce au module précédent qu'il vous est finalement inutile ou qu'il est entièrement à repenser.

```
CLS
PRINT STRING$(&50,"-")
LOCATE 15,3:PRINT"          S U P P R I M E R   L E   T E X T E"
PRINT STRING$(&50,"-")
LOCATE 5,10:INPUT"Etes-vous certain de vouloir supprimer le texte";choix
IF LEFT$(choix$,1)="o" THEN CLEAR:GOTO 1070
GOTO 100
RUN
```

Explication du programme:

-------------------------

1010 Vidage de l'écran

1020-1040 En-tête du module

1050 Vous devez confirmer votre décision de supprimer le texte

1060 La variable CLEAR supprime le texte si vous avez répondu "o". Le programme saute en 1070

1065 Si vous ne voulez plus supprimer votre texte, le programme saute au menu

1070 C'est ici que saute le programme si vous avez supprimé votre texte. Le programme est relancé pour que soit redimensionnée la variable contenu$ qui a été détruite par l'instruction CLEAR

Variables employées:

--------------------

choix$ Décision oui/non

Améliorations possibles:

------------------------

Il n'y pas ici d'améliorations essentielles à apporter si ce n'est en ce qui concerne la présentation. La possibilité de supprimer uniquement des parties de texte serait également intéressante mais relève plutôt du module modification.

Mode d'emploi de ce module:

---------------------------

Vous pouvez confirmer votre décision de supprimer le texte en entrant "o" ou retourner au menu en entrant "n".

MODULE 6 -SAUVEGARDER LE TEXTE-

Ce module vous permet de sauvegarder votre texte sur cassette. Il s'agit là d'un des éléments principaux du programme car la possibilité de conserver des textes sous cette forme constitue le principal avantage du traitement de texte par rapport à la machine à écrire. Vous pouvez en effet, de ce fait, tirer autant d'exemplaires que vous voulez de votre texte quand vous le voulez. Vous pouvez conserver des modèles de lettres types (factures, changement d'adresse, curriculum vitae, etc...) que vous pouvez ensuite adapter très aisément à différents correspondants, etc... Le traitement de texte constitue d'autre part un mode d'archivage très économe en place.

Explication du programme:
-------------------------

| | |
|---|---|
| 1080 | Vidage de l'écran |
| 1090-1100 | En-tête du module |
| 1120 | Message à l'utilisateur |
| 1130 | Le programme attend que vous appuyiez sur une touche |
| 1140 | La vitesse d'écriture est augmentée pour que la sauvegarde ne prenne pas trop de temps. Si vous n'utilisez pas des cassettes de bonne qualité il vaudrait mieux supprimer cette instruction de façon à ce que vos textes ne soient pas perdus. Il n'est cependant pas recommandé d'utiliser d'une manière générale des cassettes de mauvaise qualité. Le mieux est d'éviter d'acheter les cassettes les plus chères ou les moins chères et de s'en tenir à des cassettes audio ordinaires ou bien à des cassettes spéciales informatique |

1150 Ouverture d'un fichier sur cassette avec le nom du texte se trouvant en mémoire

1160-1170 Boucle d'appel de tous les emplacements de l'écran

1180 La variable a$ stocke successivement tous les caractères composant le texte

1190 Le contenu de la variable est écrit sur la bande

1200-1210 Fermeture de la boucle

1220 Fermeture du fichier

Variables employées:
-------------------

ligne Comme précédemment

colonne Comme précédemment

a$ Comme précédemment

contenu$ Comme précédemment

nomtexte$ Nom du texte

Améliorations possibles:
------------------------

Il n'y a pas ici d'améliorations essentielles à apporter. Vous pouvez toutefois améliorer la présentation et l'ergonomie de ce module. Vous pourriez également permettre à l'utilisateur de choisir entre les deux vitesses de stockage du texte et lui demander, une fois la sauvegarde effectuée s'il souhaite procéder à une seconde sauvegarde.

Mode d'emploi de ce module:
---------------------------

Une fois que vous avez choisi ce module, vous devez appuyer sur une touche. Le programme ouvre alors un fichier d'écriture sur la

bande, ce qui dure un certain temps. Le message vous invitant à appuyer sur les touches PLAY et RECORD du lecteur de cassette apparaît peu après. Vous n'avez plus ensuite qu'à appuyer sur une touche quelconque pour que la sauvegarde s'effectue.

```
1080 CLS
1090 PRINT STRING$(&50,"-")
1100 LOCATE 15,3:PRINT"          S A U V E G A R D E R   L E   T E X T E"
1110 PRINT STRING$(&50,"-")
1120 LOCATE 10,10:PRINT"Frappez une touche pour sauvegarder le texte"
1130 CALL &BB18
1140 SPEED WRITE 1
1150 OPENOUT nomtexte$
1160 FOR ligne=1 TO 24
1170 FOR colonne=1 TO 80
1180 a$=contenu$(colonne,ligne)
1190 PRINT£9,a$
1200 NEXT colonne
1210 NEXT ligne
1220 CLOSEOUT
1230 GOTO 100
```

MODULE 7 -CHARGER UN TEXTE-

Ce module vous permet de charger les textes que vous avez sauvegardés sur cassette.

```
1240 CLS
1250 PRINT STRING$(&50,"-")
1260 PRINT"                    C H A R G E R       L E       T E X T E"
1270 PRINT STRING$(&50,"-")
1280 LOCATE 10,10:PRINT"Frappez une touche pour charger le texte"
1290 CALL &BB18
1300 OPENIN nomtexte$
1310 FOR ligne=1 TO 24
1320 FOR colonne=1 TO 80
1330 INPUT£9,a$
1340 contenu$(colonne,ligne)=a$
1350 NEXT colonne
1360 NEXT ligne
1370 CLOSEIN
1380 GOTO 100
```

Explication du programme:

-------------------------

1240 Vidage de l'écran

1250-1270 En-tête du module

1280 Message à l'utilisateur

1290 Le programme attend que vous appuyiez sur une touche

1300 Ouverture d'un fichier sur cassette en lecture

1330 Boucle de lecture des données sur la bande

1340 Les données lues dans a$ sont affectées à la variable

contenu$ correspondant au bon emplacement sur l'écran

1350-1360 Fermeture des boucles

1370 Fermeture du fichier

1380 Retour au menu

Variables employées:

---------------------

ligne Comme précédemment

colonne Comme précédemment

a$ Comme précédemment

contenu$ Comme précédemment

Améliorations possibles:

-------------------------

Il n'y a pas ici d'améliorations importantes à apporter.

Mode d'emploi de ce module:

----------------------------

Le message vous invitant à appuyer sur la touche PLAY du lecteur de cassette apparaît peu après que vous ayez sélectionné ce module et appuyé sur une touche. Enroulez la cassette jusqu'au début du texte à charger puis appuyez sur la touche PLAY. Vous n'avez plus ensuite qu'à appuyer sur une touche quelconque pour que le chargement s'effectue.

MODULE 8 -IMPRIMER LE TEXTE-

Ce module vous permet de faire imprimer votre texte.

```
1390 CLS
1400 PRINT STRING$(&50,"-")
1410 PRINT"                    I M P R I M E R -     L E      T E X T E"
1420 PRINT STRING$(&50,"-")
1430 LOCATE 10,10:PRINT"Frappez une touche pour imprimer le texte"
1440 CALL &BB18
1450 FOR ligne=1 TO 24
1460 FOR colonne=1 TO 80
1470 a$=contenu$(colonne,ligne)
1480 PRINT£8,a$;
1490 NEXT colonne
1495 PRINT £8,CHR$(13);CHR$(10);
1500 NEXT ligne
1510 NEXT champs
1520 NEXT enregis
1530 GOTO 100
```

Explication du programme:

------------------------

1390 Vidage de l'écran

1400-1420 En-tète du module

1430 Message à l'utilisateur

1440 Le programme attend que vous appuyiez sur une touche

1450-1460 Boucle d'impression du texte

1470 La variable a$ reçoit successivement tous les caractères à imprimer

1480 Impression d'un caractère

1490-1500 Fermeture des boucles

1495 A la fin de chaque ligne, la tête d'impression est ramenée en début de ligne

1530 Retour au menu

Variables employées:

---------------------

| | |
|---|---|
| ligne | Comme précédemment |
| colonne | Comme précédemment |
| contenu$ | Comme précédemment |
| a$ | Comme précédemment |
| choix$ | Comme précédemment |

Améliorations possibles:

------------------------

La routine actuelle est relativement simple: les lignes du texte sont imprimées les unes après les autres sans aucun format, telles que vous les avez entrées.
Une impression formatée vous permettrait par exemple de faire justifier votre texte, c'est-à-dire de faire rajouter automatiquement des espaces à l'intérieur des lignes de façon à ce que les bords verticaux de votre texte apparaissent alignés et non en dent de scie. C'est ainsi que le texte que vous lisez a été imprimé.

Ceci n'est malheureusement pas très facile à programmer et d'autre part très long en BASIC. Nous conseillons donc uniquement aux programmeurs expérimentés de s'y attaquer en créant de préférence une routine en langage-machine à cet effet.

Mode d'emploi de ce module:

---------------------------

Une fois que vous avez appelé ce module, vous devez connecter l'imprimante à votre ordinateur et appuyer sur une touche.
L'impression une fois effectuée vous n'avez plus qu'à indiquer si vous voulez une seconde impression ou si vous voulez revenir au menu.

MODULE 9 -FIN DU PROGRAMME-

Ce module vous permet de mettre fin au programme.

```
1540 cls
1550 print string$(&50,"-")
1560 print"         F I N   D U   P R O G R A M M E"
1570 print string$(&50,"-")
1580 locate 5,10:input"Confirmez (o/n)";choix$
1590 if choix$="o" then end else goto 100
```

Explication du programme:

------------------------

1540 Vidage de l'écran

1550-1570 En-tête du module

1580 On vous demande de confirmer que vous voulez bien mettre fin au programme. Si "non", vous retournez au menu

Variables employées:

--------------------

choix$ Décision oui/non

Améliorations possibles:

-----------------------

Les seules améliorations possibles concernent la présentation.

Mode d'emploi de ce module:

--------------------------

Le mode d'emploi de ce module ne présente pas non plus de difficulté, il vous suffit soit de confirmer que vous voulez mettre fin au programme, soit d'indiquer que vous souhaitez

revenir au menu. N'oubliez pas que si vous mettez fin au programme, tout texte en mémoire que vous n'auriez pas sauvegardé sur cassette sera détruit.

7.4 ATTRAPEZ LA BOMBE

Comme vous n'allez certainement pas utiliser votre CPC 464 uniquement pour de difficiles travaux, nous avons pensé que vous seriez heureux de trouver les deux jeux que nous vous fournissons en guise de conclusion de cet ouvrage. Notre premier jeu s'appelle "attrapez la bombe" et son principe est tellement simple que vous le comprendrez dès l'abord mais vous verrez vite qu'il n'est pas aussi facile de gagner que vous pourriez le penser à première vue.

Il s'agit d'attraper avec un wagonnet les bombes qui tombent du ciel. Si vous attrapez la bombe vous marquez un point, sinon vous perdez un point. Vous pouvez déterminer au début du jeu la vitesse de chute des bombes et le nombre de bombes que vous allez devoir attraper.
Vous déplacez le wagonnet avec les touches curseur gauche et droite.

Faites bien attention en ligne 180 à bien entrer deux espaces après l'instruction PRINT.

La meilleure vitesse pour commencer est 0.5. Mais si vous aimez vivre dangeureusement, vous pouvez choisir une vitesse supérieure à 2.

Il vaut mieux ne pas interrompre le jeu avec la touche ESC car nous avons diminué le délai de répétition des touches et vous aurez beaucoup de mal à contrôler votre ordinateur. Attendez plutôt la fin d'une partie.

Encore un conseil pour les lecteurs très courageux: si vous modifiez la ligne 160 ainsi:

```
160 d=1:anf=int(rnd(1)*30)
```

et la ligne 260 ainsi:

```
260 IF ko>30 THEN ko=30
```

vous verrez que les bombes tomberont sur une encore plus grande largeur horizontale ce qui rend leur "capture" encore plus difficile.

Vous pouvez bien sûr apporter autant de modifications que vous le souhaitez à ce petit programme, notamment en améliorant le graphisme de l'arrière-plan ou en ajoutant des effets sonores. Il serait également intéressant de créer des caractères redéfinis vous permettant de montrer la bombe qui explose. Vous pourriez alors déplacer ces caractères sur l'écran avec l'instruction LOCATE lorsque la bombe n'a pas été attrapée.

```
10 REM attrappez la bombe
20 MODE 1
30 SYMBOL AFTER 57
40 SYMBOL 58,128,192,192,255,255,201,28,8
50 SYMBOL 59,1,3,3,255,255,147,58,16
60 SYMBOL 60,4,8,28,62,127,127,62,28
70 SYMBOL 61,255,227,255,255,255,255,255,255
80 INPUT"vitesse de chute";fa
90 INPUT"Nombre de bombes";bo
100 SPEED KEY 1,1
110 CLS
120 ko=10
130 FOR bombes=1 TO bo
140 CLS
150 LOCATE 1,22:PRINT STRING$(&28,"=")
160 d=1:anf=INT(RND(1)*20)+1
170 LOCATE anf,d:PRINT" "
180 LOCATE ko,21:PRINT":;"
190 LOCATE anf,d+fa:PRINT"<"
200 LOCATE ko,21:PRINT"  "
210d=d+fa
220 IF INT(d)>=20 THEN GOTO 280
230 in$=INKEY$
240 ko=ko+(1 AND in$=CHR$(243))-(1 AND in$=CHR$(242))
250 IF ko<1 THEN ko=1
260 IF ko>25 THEN ko=25
270 GOTO 170
280 IF ko=anf OR anf=ko+1 THEN tr=tr+1
290 IF ko<>anf AND ko<>anf-1 THEN tr=tr-1
300 NEXT
310 CLS
320 SPEED KEY 20,3
330 LOCATE 15,10:PRINT"SCORE-",tr
340 LOCATE 10,20:PRINT"encore une fois?"
350 INPUT a$
360 IF a$="o" THEN RUN
370 SPEED KEY 20,3
```

7.5 LA BATAILLE NAVALE

Voici le deuxième jeu que nous vous avons concocté. Contrairement à ce que le titre pourrait faire penser, il ne s'agit pas de la millième version de ce jeu très connu.

Vous devez tout d'abord entrer certains paramètres:

- le niveau de difficulté du jeu qui doit être compris entre 0 et 2. Les niveaux 0, 1 et 2 correspondent à des écrans de 20, 40 ou 80 colonnes. Le niveau 0 est le niveau le plus agréable sur le plan de la qualité du graphisme mais c'est bien sûr aussi le plus facile.

- l'unité de temps. Il s'agit du temps dont vous disposez pour exécuter votre tâche. Plus ce paramètre est petit, plus votre tâche sera difficile car vous disposerez de moins de temps pour l'accomplir.

- la vitesse de déplacement de la bombe sur l'écran

- le nombre de bombes avec lesquelles vous voulez jouer

Il est maintenant temps que nous vous expliquions le but du jeu:

Une fois que vous avez entré les paramètres ci-dessus, l'image du jeu apparaît. Vous voyez d'une part un bateau ennemi avec un canon, d'autre part un sous-marin et une bombe à côté de ce sous-marin. Votre tâche consiste à déplacer la bombe avec les touches curseur et à l'amener sur le pont du bateau ennemi pour détruire

celui-ci. Si vous détruisez le bateau dans le temps qui vous est imparti, vous marquez des points, sinon l'écran de jeu suivant apparaît.

A la fin du jeu apparaît une notation de votre prestation qui tient compte du nombre de bateaux touchés, du temps que vous avez mis, du niveau de difficulté et de la vitesse de la bombe.

Vous pouvez bien sûr améliorer ce jeu à votre guise en améliorant le graphisme et les effets sonores. Vous pourrez certainement puiser de précieuses idées dans les chapitres 2 et 3.

```
10 MODE 2
15 touches=0
20 SYMBOL AFTER 32
30 LOCATE 20,10:PRINT"B A T A I L L E   N A V A L E"
40 GOSUB 350
50 LOCATE 1,15:INPUT"Quel niveau de difficulte (0-20)";a
60 INPUT"Combien d'unites de temps pour diriger une bombe";b
70 INPUT"Vitesse de la bombe";vitesse
80 INPUT"Combien de bombes";bombes
85 DIM bo1(bombes)
90 MODE a
95 FOR bb=1 TO bombes
96 CLS
100 IF a=0 THEN f=18
110 IF a=1 THEN f=38
120 IF a=2 THEN f=78
130 yp=INT(RND(1)*24)+2
140 xp=INT(RND(1)*f)+1
150 yp2=INT(RND(1)*25)+1
160 xp2=INT(RND(1)*f)+1
170 IF xp=xp2 OR yp=yp2 THEN GOTO 130
171 xp3=xp2+2:yp3=yp2
172 b1=b*100
175 SPEED KEY 1,vitesse
176 FOR mnn=1 TO b1
180 LOCATE xp+2,yp-1:PRINT"!"
190 LOCATE xp,yp:PRINT"&%£"
200 LOCATE xp2,yp2:PRINT"][" 
220 LOCATE xp3,yp3:PRINT" "
230 in$=INKEY$
240 xp3=xp3+(1 AND in$=CHR$(243))-(1 AND in$=CHR$(242))
250 yp3=yp3+(1 AND in$=CHR$(241))-(1 AND in$=CHR$(240))
260 IF xp3<1 THEN xp3=1
270 IF xp3>f+2 THEN xp3=f+2
280 IF yp3<1 THEN yp3=1
290 IF yp3>25 THEN yp3=25
300 IF yp3=25 AND xp3=f+2 THEN xp3=f+1
310 LOCATE xp3,yp3:PRINT"^"
320 IF xp3=xp AND yp3=yp THEN GOTO 440
330 NEXT mnn
335 NEXT bb
340 GOTO 3000
```

```
350 REM definir un caractere
360 SYMBOL 33,62,8,28,28,124,124,124,252
370 SYMBOL 35,252,252,252,255,254,252,248,240
380 SYMBOL 37,1,129,129,255,255,255,255,255
390 SYMBOL 38,126,15,15,127,63,15,3,1
400 SYMBOL 91,128,192,252,254,255,255,254,0
410 SYMBOL 93,0,1,31,63,255,255,63,0
420 SYMBOL 94,0,0,156,190,127,190,156,0
430 RETURN
440 MODE 0
445 touches=touches+1
446 LOCATE 3,12
447 bo1(bb)=((b1-mnn)/b1)*10
448 FOR zei=1 TO 13
449 PRINT CHR$(224);
450 NEXT
451 LOCATE 3,13:PRINT CHR$(224);"T O U C H E";CHR$(224)
452 LOCATE 3,14
453 FOR zei=1 TO 13
454 PRINT CHR$(224);
455 NEXT
460 SOUND 129,450,150,15,,,15
465 FOR xx=1 TO 3000
466 NEXT xx
467 MODE a
470 GOTO 335
3000 MODE 2
3005 FOR II=1 TO bombes
3006 Points=Points+bo1(II)
3007 NEXT
3008 Points=Points*touches*(A+1)
3010 LOCATE 15,9:PRINT"L E  J E U  E S T  F I N I"
3020 LOCATE 2,11:PRINT"Vous avez fait";touches;"touches sur";bombes;"bombes;'
3021 LOCATE 2,13:PRINT"VOTRE SCORE EST:";INT(Points)
3025 SPEED KEY 20,3
3030 INPUT"Voulez-vous jouer encore une fois";choix$
3040 IF choix$="o" THEN RUN
3050 END
```

## LISTE DES PRODUITS
## MICRO APPLICATION POUR CPC 464

| *LIVRES* | *PRIX TTC* |
|---|---|
| — Trucs et Astuces | 149 FF |
| — PROGRAMMES BASIC | 149 FF |
| — LES JEUX D'AVENTURES (et comment les programmer) | 129 FF |
| — ANATOMIE DU CPC 464 | 179 FF |

*LOGICIELS*

| | |
|---|---|
| — GESTION DE FICHIER K7 | 245 FF |
| — SERIE NOIRE (jeu de rôle) | 145 FF |
| — ISLAND (jeu éducatif) | 120 FF |
| — LES JEUX DE MAITRE JACQUES (jeu éducatif) | 120 FF |
| — CHATEAU HANTE (jeu d'aventure) | 120 FF |

Achevé d'imprimer en janvier 1985
sur les presses de l'imprimerie Laballery et Cie
58500 Clamecy
Dépôt légal : janvier 1985
N° d'imprimeur : 501078

IMPRIMÉ EN FRANCE

[redacted]

Ce livre vous dévoile de nombreux trucs et astuces pour l'AMSTRAD CPC 464.

THÈMES DÉVELOPPÉS :

- Structure Hardware
- Système d'exploitation
- Tokens BASIC
- Dessin avec joystick
- Technique et applications des fenêtres
- Programme complet de gestion fichier
- Editeur de son
- Générateur de caractères
- Listings détaillés de jeux passionnants

Pour tous les fanas du CPC 464.

ISBN : 2-86899-005-3
149 FF TTC

Réf. : ML 112

AMSTRAD





Document numérisé avec amour par

# AMSTRAD CPC

# MÉMOIRE ÉCRITE




https://acpc.me/